VIPA System 200V

CP | Handbuch

HB97D_CP | RD_240-1BA20 | Rev. 14/45

November 2014

VIPA
A YASKAWA COMPANY

VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH
Ohmstraße 4, D-91074 Herzogenaurach, Germany
Tel.: +49 (91 32) 744 -0
Fax.: +49 9132 744 1864
EMail: info@vipa.de
http://www.vipa.com

Hinweis

Es wurden alle Anstrengungen unternommen, um sicherzustellen, dass die in diesem Dokument enthaltenen Informationen zum Zeitpunkt der Veröffentlichung vollständig und richtig sind. Das Recht auf Änderungen der Informationen bleibt jedoch vorbehalten.

Die vorliegende Kundendokumentation beschreibt alle heute bekannten Hardware-Einheiten und Funktionen. Es ist möglich, dass Einheiten beschrieben sind, die beim Kunden nicht vorhanden sind. Der genaue Lieferumfang ist im jeweiligen Kaufvertrag beschrieben.

EG-Konformitätserklärung

Hiermit erklärt VIPA GmbH, dass die Produkte und Systeme mit den grundlegenden Anforderungen und den anderen relevanten Vorschriften übereinstimmen.

Die Übereinstimmung ist durch CE-Zeichen gekennzeichnet.

Informationen zur Konformitätserklärung

Für weitere Informationen zur CE-Kennzeichnung und Konformitätserklärung wenden Sie sich bitte an Ihre Landesvertretung der VIPA GmbH.

Warenzeichen

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S und Commander Compact sind eingetragene Warenzeichen der VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SPEED7 ist ein eingetragenes Warenzeichen der profichip GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300 und S7-400 sind eingetragene Warenzeichen der Siemens AG.

Microsoft und Windows sind eingetragene Warenzeichen von Microsoft Inc., USA.

Portable Document Format (PDF) und Postscript sind eingetragene Warenzeichen von Adobe Systems, Inc.

Alle anderen erwähnten Firmennamen und Logos sowie Marken- oder Produktnamen sind Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer.

Dokument-Support

Wenden Sie sich an Ihre Landesvertretung der VIPA GmbH, wenn Sie Fehler anzeigen oder inhaltliche Fragen zu diesem Dokument stellen möchten. Ist eine solche Stelle nicht erreichbar, können Sie VIPA über folgenden Kontakt erreichen:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax: +49 9132 744 1204
EMail: documentation@vipa.de

Technischer Support

Wenden Sie sich an Ihre Landesvertretung der VIPA GmbH, wenn Sie Probleme mit dem Produkt haben oder Fragen zum Produkt stellen möchten. Ist eine solche Stelle nicht erreichbar, können Sie VIPA über folgenden Kontakt erreichen:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefon: +49 9132 744 1150 (Hotline)
EMail: support@vipa.de

Inhaltsverzeichnis

Über dieses Handbuch ... 1
Sicherheitshinweise ... 2
Teil 1 Grundlagen und Montage ... 1-1
Sicherheitshinweis für den Benutzer ... 1-2
Systemvorstellung ... 1-3
Abmessungen ... 1-5
Montage ... 1-7
Demontage und Modultausch ... 1-11
Verdrahtung ... 1-12
Aufbaurichtlinien ... 1-14
Allgemeine Daten ... 1-17
Teil 2 Hardwarebeschreibung ... 2-1
Leistungsmerkmale ... 2-2
Aufbau ... 2-3
Technische Daten ... 2-6
Teil 3 Einsatz ... 3-1
Schnelleinstieg ... 3-2
GSD und FCs einbinden ... 3-4
Projektierung ... 3-5
Standardhantierungsbausteine ... 3-8
RK512-Kommunikation - Hantierungsbausteine ... 3-13
RK512-Kommunikation - Anzeigewort ANZW ... 3-18
ASCII / STX/ETX / 3964(R) / RK512 - Grundlagen ... 3-20
ASCII / STX/ETX / 3964(R) / RK512 - Kommunikationsprinzip ... 3-26
ASCII / STX/ETX / 3964(R) / RK512 - Parametrierung ... 3-29
Modbus - Grundlagen ... 3-36
Modbus - Parametrierung ... 3-38
Modbus - Einsatz ... 3-41
Modbus - Funktionscodes ... 3-45
Modbus - Fehlermeldungen ... 3-49
Modbus - Beispiel ... 3-50

Über dieses Handbuch

Das Handbuch beschreibt den bei VIPA erhältlichen System 200V CP 240-1BA20. Hier finden Sie eine detaillierte Beschreibung des CPs. Sie erhalten Informationen für den Anschluss und die Handhabe des CPs im System 200V und die Technischen Daten des Moduls.

Überblick

Teil 1: Grundlagen und Montage

Kernthema dieses Kapitels ist die Vorstellung des System 200V von VIPA. Hier finden Sie alle Informationen, die für den Aufbau und die Verdrahtung einer Steuerung aus den Komponenten des System 200V erforderlich sind. Neben den Abmessungen sind hier auch die allgemeinen technischen Daten des System 200V aufgeführt.

Teil 2: Hardwarebeschreibung

In diesem Kapitel finden Sie Informationen über den Aufbau und die Anschlussbelegung des Kommunikationsprozessors CP 240 mit RS232-Schnittstelle.

Teil 3: Einsatz

Den Kommunikationsprozessor CP 240 erhalten Sie von VIPA mit verschiedenen Übertragungsprotokollen, auf deren Einsatz hier näher eingegangen wird.

Zielsetzung und Inhalt

Das Handbuch beschreibt den CP 240-1BA20 aus dem System 200V von VIPA. Beschrieben wird Aufbau, Projektierung und Anwendung.

Dieses Handbuch ist Bestandteil des Dokumentationspakets mit der Best.-Nr.: HB97D_CP und gültig für:

Produkt	Best.-Nr.	ab Stand: HW
CP 240 RS232	VIPA CP 240-1BA20	01

Zielgruppe

Das Handbuch ist geschrieben für Anwender mit Grundkenntnissen in der Automatisierungstechnik.

Aufbau des Handbuchs

Das Handbuch ist in Kapitel gegliedert. Jedes Kapitel beschreibt eine abgeschlossene Thematik.

Orientierung im Dokument

Als Orientierungshilfe stehen im Handbuch zur Verfügung:

- Gesamt-Inhaltsverzeichnis am Anfang des Handbuchs
- Übersicht der beschriebenen Themen am Anfang jedes Kapitels

Verfügbarkeit

Das Handbuch ist verfügbar in:

- gedruckter Form auf Papier
- in elektronischer Form als PDF-Datei (Adobe Acrobat Reader)

Piktogramme Signalwörter

Besonders wichtige Textteile sind mit folgenden Piktogrammen und Signalworten ausgezeichnet:

Gefahr!

Unmittelbar drohende oder mögliche Gefahr.

Personenschäden sind möglich.

Achtung!

Bei Nichtbefolgen sind Sachschäden möglich.

Hinweis!

Zusätzliche Informationen und nützliche Tipps

Sicherheitshinweise

Bestimmungs-gemäße Verwendung

Der CP 240 ist konstruiert und gefertigt für:

- alle VIPA System-200V-Komponenten
- Kommunikation und Prozesskontrolle
- Allgemeine Steuerungs- und Automatisierungsaufgaben
- den industriellen Einsatz
- den Betrieb innerhalb der in den technischen Daten spezifizierten Umgebungsbedingungen
- den Einbau in einen Schaltschrank

Gefahr!

Das Gerät ist nicht zugelassen für den Einsatz

- in explosionsgefährdeten Umgebungen (EX-Zone)

Dokumentation

Handbuch zugänglich machen für alle Mitarbeiter in

- Projektierung
- Installation
- Inbetriebnahme
- Betrieb

Vor Inbetriebnahme und Betrieb der in diesem Handbuch beschriebenen Komponenten unbedingt beachten:

- Hardware-Änderungen am Automatisierungssystem nur im spannungslosen Zustand vornehmen!
- Anschluss und Hardware-Änderung nur durch ausgebildetes Elektro-Fachpersonal
- Nationale Vorschriften und Richtlinien im jeweiligen Verwenderland beachten und einhalten (Installation, Schutzmaßnahmen, EMV ...)

Entsorgung

Zur Entsorgung des Geräts nationale Vorschriften beachten!

Teil 1 Grundlagen und Montage

Übersicht

Kernthema dieses Kapitels ist die Vorstellung des System 200V von VIPA. Hier finden Sie alle Informationen, die für den Aufbau und die Verdrahtung einer Steuerung aus den Komponenten des System 200V erforderlich sind.

Neben den Abmessungen sind hier auch die allgemeinen technischen Daten des System 200V aufgeführt.

Inhalt

Thema	Seite
Teil 1 Grundlagen und Montage	**1-1**
Sicherheitshinweis für den Benutzer	1-2
Systemvorstellung	1-3
Abmessungen	1-5
Montage	1-7
Demontage und Modultausch	1-11
Verdrahtung	1-12
Aufbaurichtlinien	1-14
Allgemeine Daten	1-17

Sicherheitshinweis für den Benutzer

Handhabung elektrostatisch gefährdeter Baugruppen

VIPA-Baugruppen sind mit hochintegrierten Bauelementen in MOS-Technik bestückt. Diese Bauelemente sind hoch empfindlich gegenüber Überspannungen, die z.B. bei elektrostatischer Entladung entstehen.

Zur Kennzeichnung dieser gefährdeten Baugruppen wird nachfolgendes Symbol verwendet:

Das Symbol befindet sich auf Baugruppen, Baugruppenträgern oder auf Verpackungen und weist so auf elektrostatisch gefährdete Baugruppen hin.

Elektrostatisch gefährdete Baugruppen können durch Energien und Spannungen zerstört werden, die weit unterhalb der Wahrnehmungsgrenze des Menschen liegen. Hantiert eine Person, die nicht elektrisch entladen ist, mit elektrostatisch gefährdeten Baugruppen, können Spannungen auftreten und zur Beschädigung von Bauelementen führen und so die Funktionsweise der Baugruppen beeinträchtigen oder die Baugruppe unbrauchbar machen. Auf diese Weise beschädigte Baugruppen werden in den wenigsten Fällen sofort als fehlerhaft erkannt. Der Fehler kann sich erst nach längerem Betrieb einstellen.

Durch statische Entladung beschädigte Bauelemente können bei Temperaturänderungen, Erschütterungen oder Lastwechseln zeitweilige Fehler zeigen.

Nur durch konsequente Anwendung von Schutzeinrichtungen und verantwortungsbewusste Beachtung der Handhabungsregeln lassen sich Funktionsstörungen und Ausfälle an elektrostatisch gefährdeten Baugruppen wirksam vermeiden.

Versenden von Baugruppen

Verwenden Sie für den Versand immer die Originalverpackung.

Messen und Ändern von elektrostatisch gefährdeten Baugruppen

Bei Messungen an elektrostatisch gefährdeten Baugruppen sind folgende Dinge zu beachten:

- Potentialfreie Messgeräte sind kurzzeitig zu entladen.
- Verwendete Messgeräte sind zu erden.

Bei Änderungen an elektrostatisch gefährdeten Baugruppen ist darauf zu achten, dass ein geerdeter Lötkolben verwendet wird.

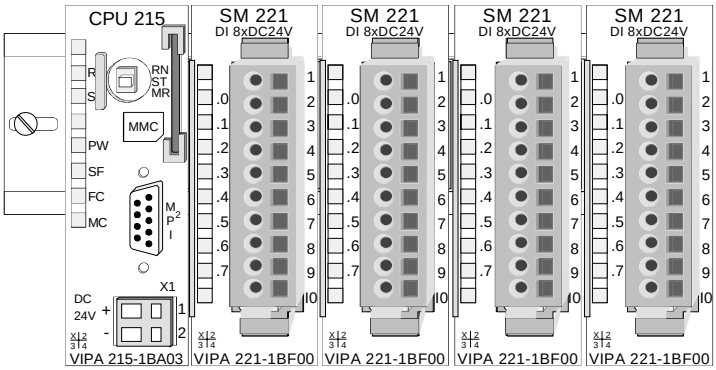
Achtung!

Bei Arbeiten mit und an elektrostatisch gefährdeten Baugruppen ist auf ausreichende Erdung des Menschen und der Arbeitsmittel zu achten.

Systemvorstellung

Übersicht

Das System 200V ist ein modular aufgebautes Automatisierungssystem für die Montage auf einer 35mm Profilschiene. Mittels der Peripherie-Module in 4-, 8- und 16-Kanalausführung können Sie dieses System passgenau an Ihre Automatisierungsaufgaben adaptieren.

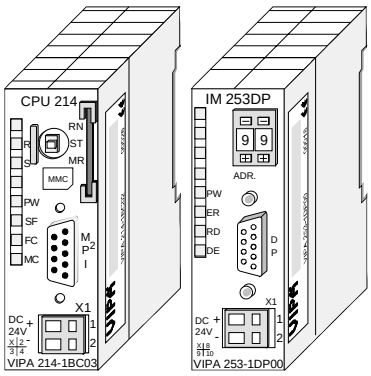


Komponenten

Das System 200V besteht aus folgenden Komponenten:

- *Kopfmodule* wie CPU und Buskoppler
- *Peripheriemodule* wie I/O-, Funktions- und Kommunikationsmodule
- *Netzteile*
- *Erweiterungsmodule*

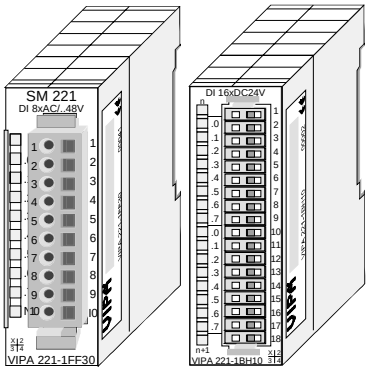
Kopfmodule



Beim Kopfmodul sind CPU bzw. Bus-Interface und DC 24V Spannungsversorgung in ein Gehäuse integriert.

Über die integrierte Spannungsversorgung werden sowohl CPU bzw. Bus-Interface als auch die Elektronik der angebunden Peripheriemodule versorgt.

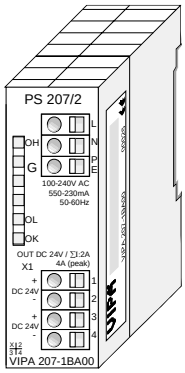
Peripheriemodule



Die einzelnen Module werden direkt auf eine 35mm-Profilschiene montiert und über Busverbinder, die vorher in die Profilschiene eingelegt werden, an das Kopfmodul gekoppelt.

Die meisten Peripheriemodule besitzen einen 10- bzw. 18poligen Steckverbinder. Über diesen Steckverbinder werden Signal- und Versorgungsleitungen mit den Modulen verbunden.

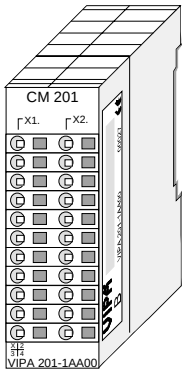
Netzteile



Die DC 24V Spannungsversorgung kann im System 200V entweder extern oder über eigens hierfür entwickelte Netzteile erfolgen.

Das Netzteil kann zusammen mit dem System 200V Modulen auf die Profilschiene montiert werden. Es besitzt keine Verbindung zum Rückwandbus.

Erweiterungsmodule



Die Erweiterungsmodule sind unter anderem Ergänzungs-Module für 2- oder 3-Draht Installation.

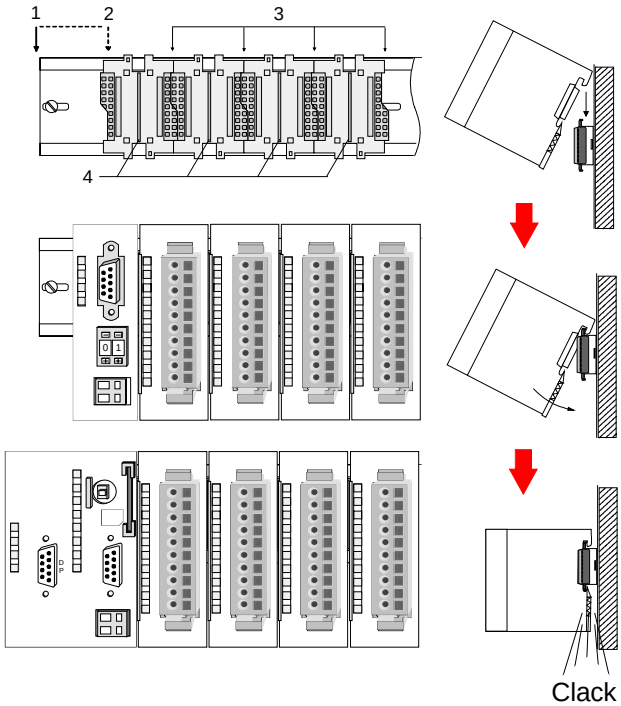
Die Module haben keine Verbindung zum Rückwandbus.

Aufbau/Maße

- Profilschiene 35mm
- Maße Grundgehäuse:
  1fach breit: (HxBxT) in mm: 76x25,4x74 in Zoll: 3x1x3
  2fach breit: (HxBxT) in mm: 76x50,8x74 in Zoll: 3x2x3

Montage

Bitte beachten Sie, dass Sie Kopfmodule nur auf Steckplatz 2 bzw. 1 und 2 (wenn doppelt breit) stecken dürfen.



[1]	Kopfmodul (doppelt breit)
[2]	Kopfmodul (einfach breit)
[3]	Peripheriemodule
[4]	Führungsleisten

Hinweis

Angaben zur maximalen Anzahl steckbarer Module und zum maximalen Strom am Rückwandbus finden Sie in den "Technischen Daten" des entsprechenden Kopfmoduls.

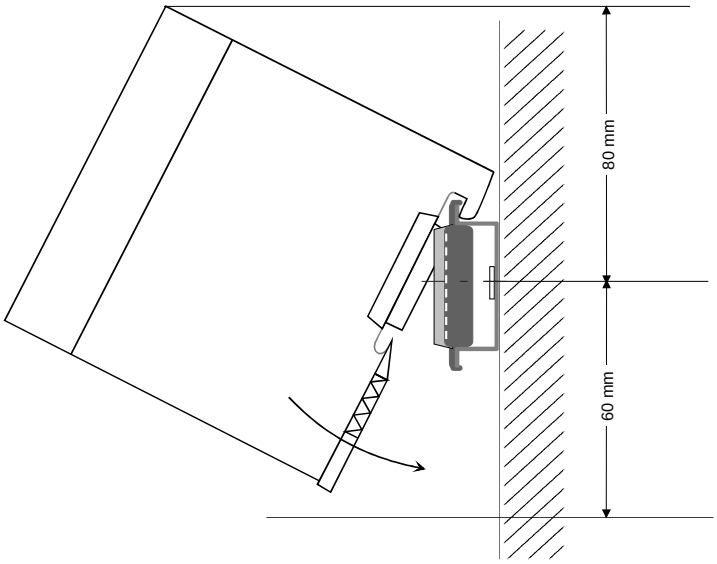
Bitte montieren Sie Module mit hoher Stromaufnahme direkt neben das Kopfmodul.

Abmessungen

Maße Grundgehäuse

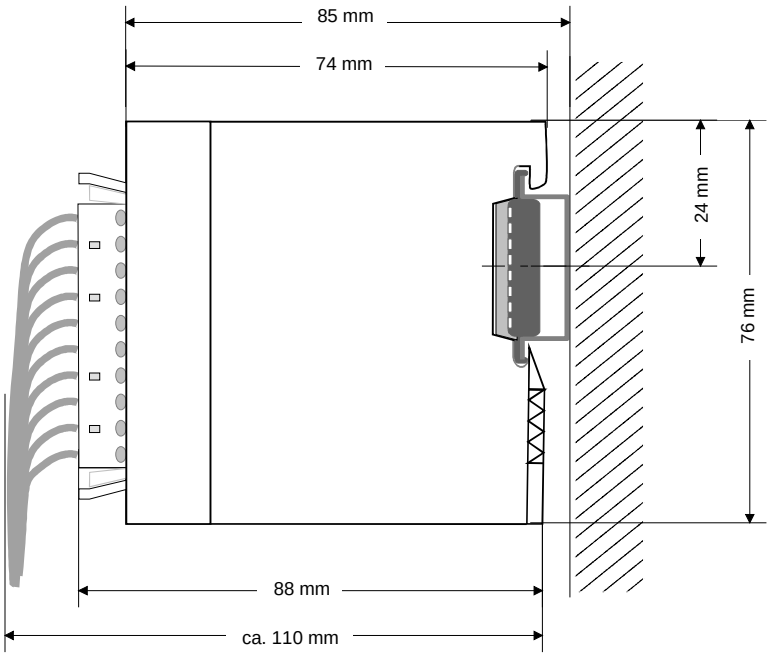
1fach breit (HxBxT) in mm: 76 x 25,4 x 74
2fach breit (HxBxT) in mm: 76 x 50,8 x 74

Montagemaße

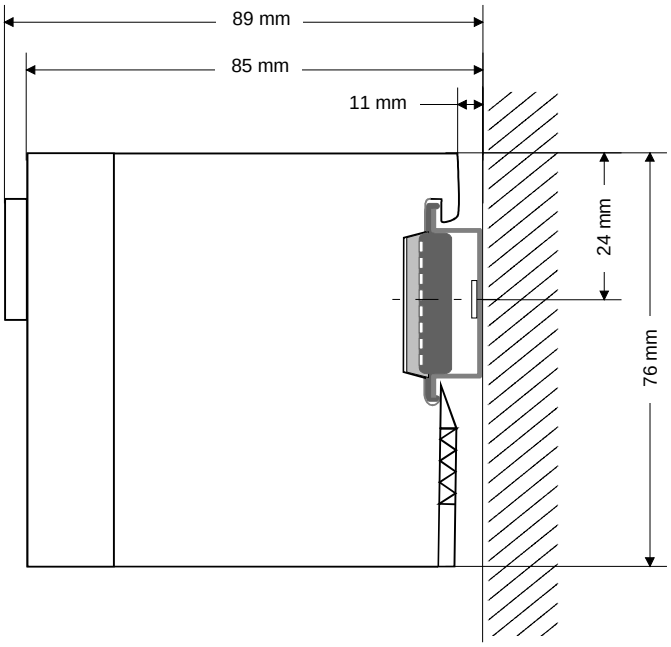


Maße montiert und verdrahtet

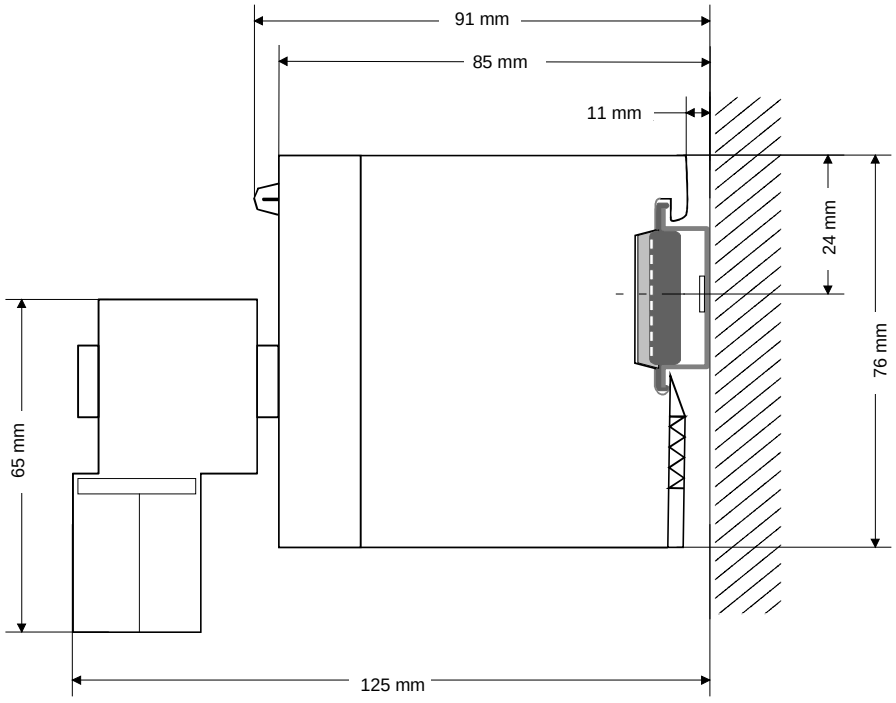
Ein- / Ausgabe-module



Funktionsmodule/
Erweiterungsmodule



CPUs (hier mit
VIPA EasyConn)



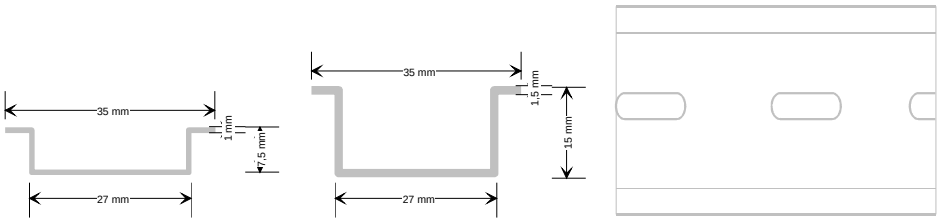
Montage

Allgemein

Die einzelnen Module werden direkt auf eine 35mm-Profilschiene montiert und über Rückwandbus-Verbinder verbunden. Vor der Montage ist der Rückwandbus-Verbinder in die Profilschiene einzulegen.

Profilschiene

Für die Montage können Sie folgende 35mm-Profilschienen verwenden:

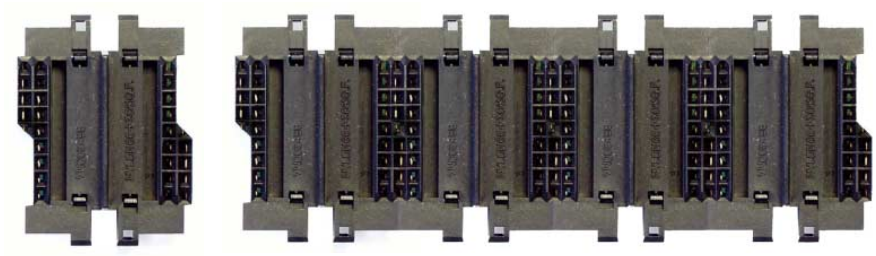


Bestellnummer	Bezeichnung	Beschreibung
290-1AF00	35mm-Profilschiene	Länge 2000mm, Höhe 15mm
290-1AF30	35mm-Profilschiene	Länge 530mm, Höhe 15mm

Busverbinder

Für die Kommunikation der Module untereinander wird beim System 200V ein Rückwandbus-Verbinder eingesetzt. Die Rückwandbusverbinder sind isoliert und bei VIPA in 1-, 2-, 4- oder 8facher Breite erhältlich.

Nachfolgend sehen Sie einen 1fach und einen 4fach Busverbinder:



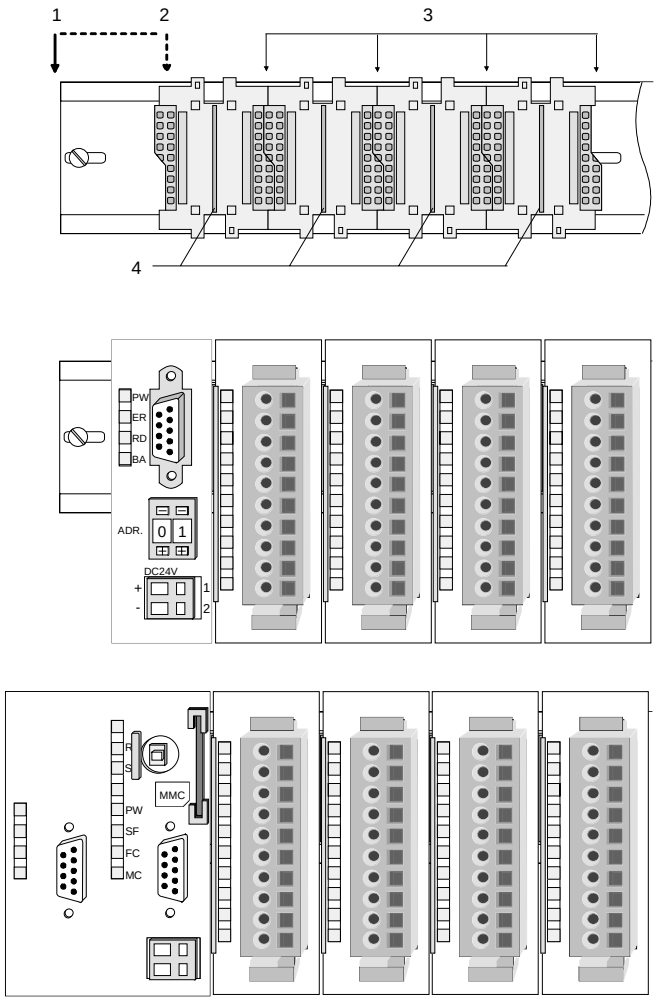
Der Busverbinder wird in die Profilschiene eingelegt, bis dieser sicher einrastet, so dass die Bus-Anschlüsse aus der Profilschiene herausschauen.

Bestellnummer	Bezeichnung	Beschreibung
290-0AA10	Busverbinder	1fach
290-0AA20	Busverbinder	2fach
290-0AA40	Busverbinder	4fach
290-0AA80	Busverbinder	8fach

Montage auf Profilschiene

Die nachfolgende Skizze zeigt einen 4fach-Busverbinder in einer Profilschiene und die Steckplätze für die Module.

Die einzelnen Modulsteckplätze sind durch Führungsleisten abgegrenzt.



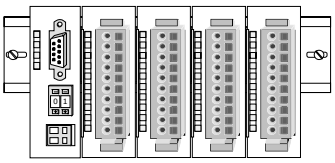
[1] Kopfmodul (doppelt breit)
[2] Kopfmodul (einfach breit)
[3] Peripheriemodule
[4] Führungsleisten

Montage unter Berücksichtigung der Stromaufnahme

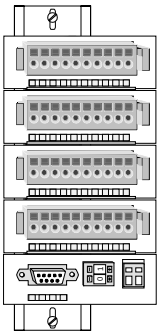
- Verwenden Sie möglichst lange Busverbinder.
- Ordnen Sie Module mit hohem Stromverbrauch direkt rechts neben Ihrem Kopfmodul an. Im Service-Bereich von www.vipa.com finden Sie alle Stromaufnahmen des System 200V in einer Liste zusammengefasst.

Montagemöglich-keiten

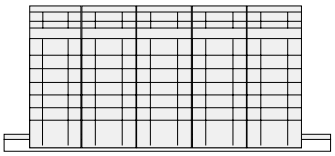
waagrechter Aufbau



senkrechter Aufbau



liegender Aufbau

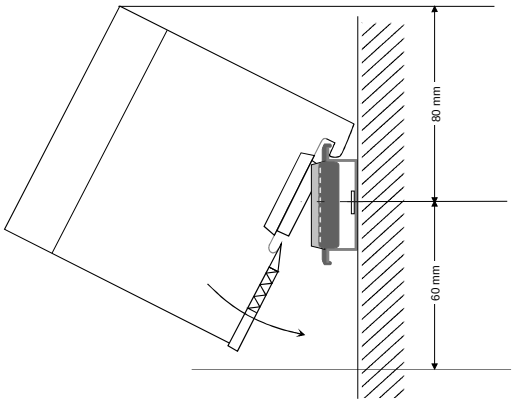


Beachten Sie bitte die hierbei zulässigen Umgebungs-temperaturen:

- waagrechter Aufbau: von 0 bis 60°C
- senkrechter Aufbau: von 0 bis 40°C
- liegender Aufbau: von 0 bis 40°C

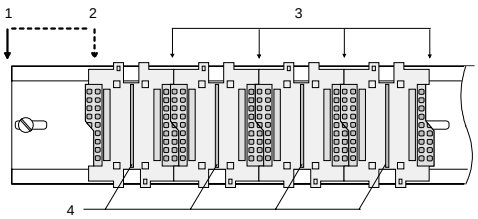
Der waagrechte Aufbau beginnt immer links mit einem Kopfmodul. Rechts daneben sind die Peripherie-Module zu stecken.

Es dürfen bis zu 32 Peripherie-Module gesteckt werden.



Bitte bei der Montage beachten!

- Schalten Sie die Stromversorgung aus bevor Sie Module stecken bzw. abziehen!
- Halten Sie ab der Mitte der Profilschiene nach oben einen Montageabstand von mindestens 80mm und nach unten von 60mm ein.



- Eine Zeile wird immer von links nach rechts aufgebaut und beginnt immer mit einem Kopfmodul.

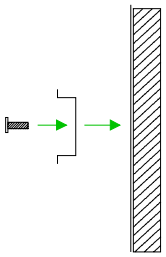
  [1] Kopfmodul (doppelt breit)
  [2] Kopfmodul (einfach breit)
  [3] Peripheriemodule
  [4] Führungsleisten

- Module müssen immer direkt nebeneinander gesteckt werden. Lücken sind nicht zulässig, da ansonsten der Rückwandbus unterbrochen ist.
- Ein Modul ist erst dann gesteckt und elektrisch verbunden, wenn es hörbar einrastet.
- Steckplätze rechts nach dem letzten Modul dürfen frei bleiben.

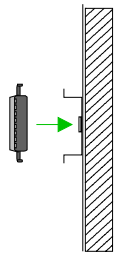
Hinweis!

Am Rückwandbus dürfen sich maximal 32 Module befinden. Hierbei darf der **Summenstrom** von **3,5A darf** nicht überschritten werden!

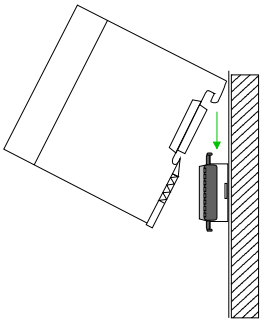
Montage
Vorgehensweise



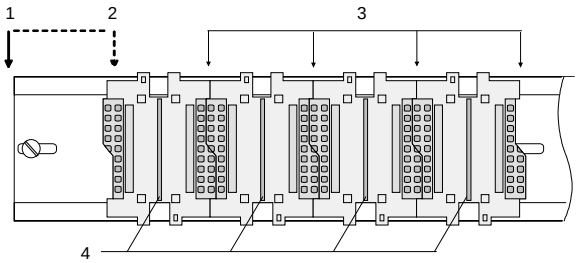
- Montieren Sie die Profilschiene. Bitte beachten Sie, dass Sie ab der Mitte der Profilschiene nach oben einen Modul-Montageabstand von mindestens 80mm und nach unten von 60mm einhalten.



- Drücken Sie den Busverbinder in die Profilschiene, bis dieser sicher einrastet, so dass die Bus-Anschlüsse aus der Profilschiene herausschauen. Sie haben nun die Grundlage zur Montage Ihrer Module.



- Beginnen Sie ganz links mit dem Kopfmodul, wie CPU, PC oder Buskoppler und stecken Sie rechts daneben Ihre Peripherie-Module.

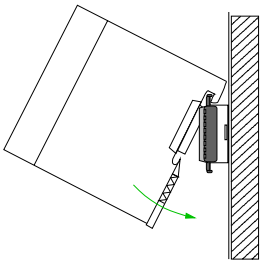


[1] Kopfmodul (doppelt breit)

[2] Kopfmodul (einfach breit)

[3] Peripheriemodule

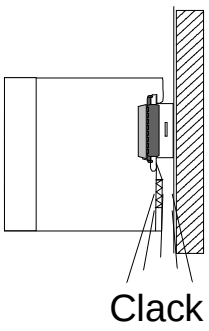
[4] Führungsleisten



- Setzen Sie das zu steckende Modul von oben in einem Winkel von ca. 45 Grad auf die Profilschiene und drehen Sie das Modul nach unten, bis es hörbar auf der Profilschiene einrastet. Nur bei eingerasteten Modulen ist eine Verbindung zum Rückwandbus sichergestellt.

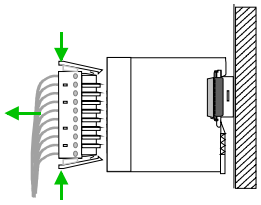
Achtung!

Module dürfen nur im spannungslosen Zustand gesteckt bzw. gezogen werden!



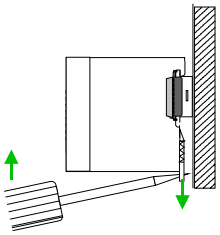
Demontage und Modultausch

1



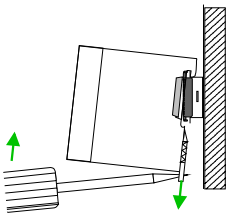
- Entfernen Sie falls vorhanden die Verdrahtung an dem Modul, indem Sie die beiden Verriegelungshebel am Steckverbinder betätigen und den Steckverbinder abziehen.

2



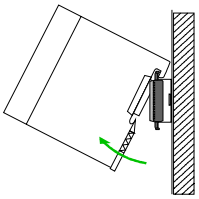
- Zur Demontage des Moduls befindet sich am Gehäuseunterteil eine gefederter Demontageschlitz. Stecken Sie, wie gezeigt, einen Schraubendreher in den Demontage-schlitz.

3



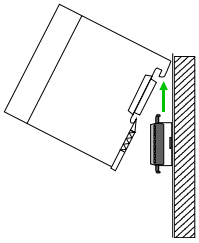
- Entriegeln Sie durch Druck des Schraubendrehers nach oben das Modul.

4



- Ziehen Sie nun das Modul nach vorn und ziehen Sie das Modul mit einer Drehung nach oben ab.

5



Achtung!

Module dürfen nur im spannungslosen Zustand gesteckt bzw. gezogen werden!

Bitte beachten Sie, dass durch die Demontage von Modulen der Rückwandbus an der entsprechenden Stelle unterbrochen wird!

Verdrahtung

Übersicht

Die meisten Peripherie-Module besitzen einen 10poligen bzw. 18poligen Steckverbinder. Über diesen Steckverbinder werden Signal- und Versorgungsleitungen mit den Modulen verbunden.

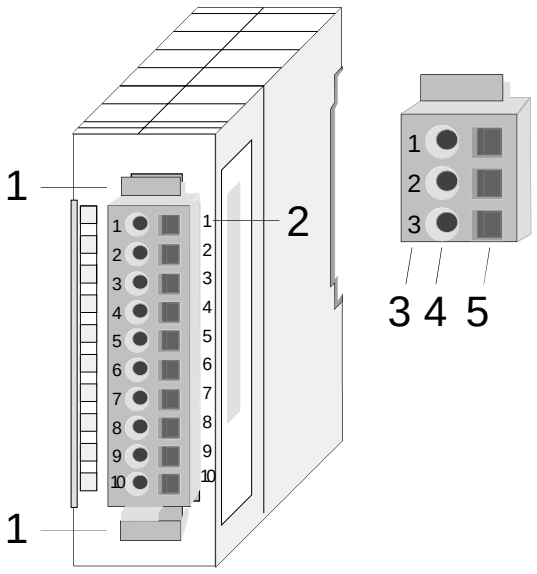
Bei der Verdrahtung werden Steckverbinder mit Federklemmtechnik eingesetzt.

Die Verdrahtung mit Federklemmtechnik ermöglicht einen schnellen und einfachen Anschluss Ihrer Signal- und Versorgungsleitungen.

Im Gegensatz zur Schraubverbindung, ist diese Verbindungsart erschütterungssicher. Die Steckerbelegung der Peripherie-Module finden Sie in der Beschreibung zu den Modulen.

Sie können Drähte mit einem Querschnitt von 0,08mm^2 bis 2,5mm^2 (bis 1,5mm^2 bei 18poligen Steckverbindern) anschließen.

Folgende Abbildung zeigt ein Modul mit einem 10poligen Steckverbinder.



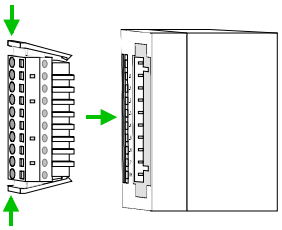
[1] Entriegelungshebel
[2] Pin-Nr. am Modul
[3] Pin-Nr. am Steckverbinder
[4] Anschluss für Draht
[5] Öffnung für Schraubendreher

Hinweis!

Die Federklemme wird zerstört, wenn Sie den Schraubendreher in die Öffnung für die Leitungen stecken!

Drücken Sie den Schraubendreher nur in die rechteckigen Öffnungen des Steckverbinders!

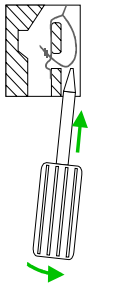
Verdrahtung
Vorgehensweise



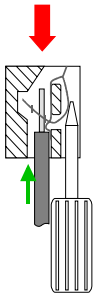
- Stecken Sie den Steckverbinder auf das Modul bis dieser hörbar einrastet. Drücken Sie hierzu während des Steckens, wie gezeigt, die beiden Verriegelungsklinken zusammen.

  Der Steckerverbinder ist nun in einer festen Position und kann leicht verdrahtet werden.

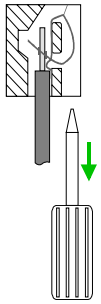
Die nachfolgende Abfolge stellt die Schritte der Verdrahtung in der Draufsicht dar.



- Zum Verdrahten stecken Sie, wie in der Abbildung gezeigt, einen passenden Schraubendreher leicht schräg in die rechteckige Öffnung.
- Zum Öffnen der Kontaktfeder müssen Sie den Schraubendreher in die entgegengesetzte Richtung drücken und halten.



- Führen Sie durch die runde Öffnung Ihren abisolierten Draht ein. Sie können Drähte mit einem Querschnitt von 0,08mm^2 bis 2,5mm^2 (bei 18poligen Steckverbindern bis 1,5mm^2) anschließen.



- Durch Entfernen des Schraubendrehers wird der Draht über einen Federkontakt sicher mit dem Steckverbinder verbunden.

Hinweis!

Verdrahten Sie zuerst die Versorgungsleitungen (Spannungsversorgung) und dann die Signalleitungen (Ein- und Ausgänge)!

Aufbaurichtlinien

Allgemeines

Die Aufbaurichtlinien enthalten Informationen über den störsicheren Aufbau von System 200V Systemen. Es werden die Wege beschrieben, wie Störungen in Ihre Steuerung gelangen können, wie die elektromagnetische Verträglichkeit (EMV), sicher gestellt werden kann und wie bei der Schirmung vorzugehen ist.

Was bedeutet EMV?

Unter Elektromagnetischer Verträglichkeit (EMV) versteht man die Fähigkeit eines elektrischen Gerätes, in einer vorgegebenen elektromagnetischen Umgebung fehlerfrei zu funktionieren ohne vom Umfeld beeinflusst zu werden bzw. das Umfeld in unzulässiger Weise zu beeinflussen.

Alle System 200V Komponenten sind für den Einsatz in rauen Industrieumgebungen entwickelt und erfüllen hohe Anforderungen an die EMV. Trotzdem sollten Sie vor der Installation der Komponenten eine EMV-Planung durchführen und mögliche Störquellen in die Betrachtung einbeziehen.

Mögliche Störeinwirkungen

Elektromagnetische Störungen können sich auf unterschiedlichen Pfaden in Ihre Steuerung einkoppeln:

- Felder
- E/A-Signalleitungen
- Bussystem
- Stromversorgung
- Schutzleitung

Je nach Ausbreitungsmedium (leitungsgebunden oder -ungebunden) und Entfernung zur Störquelle gelangen Störungen über unterschiedliche Kopplungsmechanismen in Ihre Steuerung.

Man unterscheidet:

- galvanische Kopplung
- kapazitive Kopplung
- induktive Kopplung
- Strahlungskopplung

Grundregeln zur Sicherstellung der EMV

Häufig genügt zur Sicherstellung der EMV das Einhalten einiger elementarer Regeln. Beachten Sie beim Aufbau der Steuerung deshalb die folgenden Grundregeln.

- Achten sie bei der Montage Ihrer Komponenten auf eine gut ausgeführte flächenhafte Massung der inaktiven Metallteile.
  - Stellen sie eine zentrale Verbindung zwischen der Masse und dem Erde/Schutzleitersystem her.
  - Verbinden Sie alle inaktiven Metallteile großflächig und impedanzarm.
  - Verwenden Sie nach Möglichkeit keine Aluminiumteile. Aluminium oxidiert leicht und ist für die Massung deshalb weniger gut geeignet.
- Achten Sie bei der Verdrahtung auf eine ordnungsgemäße Leitungsführung.
  - Teilen Sie die Verkabelung in Leitungsgruppen ein. (Starkstrom, Stromversorgungs-, Signal- und Datenleitungen).
  - Verlegen Sie Starkstromleitungen und Signal- bzw. Datenleitungen immer in getrennten Kanälen oder Bündeln.
  - Führen sie Signal- und Datenleitungen möglichst eng an Masseflächen (z.B. Tragholme, Metallschienen, Schrankbleche).
- Achten sie auf die einwandfreie Befestigung der Leitungsschirme.
  - Datenleitungen sind geschirmt zu verlegen.
  - Analogleitungen sind geschirmt zu verlegen. Bei der Übertragung von Signalen mit kleinen Amplituden kann das einseitige Auflegen des Schirms vorteilhaft sein.
  - Legen Sie die Leitungsschirme direkt nach dem Schrankeintritt großflächig auf eine Schirm-/Schutzleiterschiene auf, und befestigen Sie die Schirme mit Kabelschellen.
  - Achten Sie darauf, dass die Schirm-/Schutzleiterschiene impedanzarm mit dem Schrank verbunden ist.
  - Verwenden Sie für geschirmte Datenleitungen metallische oder metallisierte Steckergehäuse.
- Setzen Sie in besonderen Anwendungsfällen spezielle EMV-Maßnahmen ein.
  - Erwägen Sie bei Induktivitäten den Einsatz von Löschgliedern.
  - Beachten Sie, dass bei Einsatz von Leuchtstofflampen sich diese negativ auf Signalleitungen auswirken können.
- Schaffen Sie ein einheitliches Bezugspotential und erden Sie nach Möglichkeit alle elektrischen Betriebsmittel.
  - Achten Sie auf den gezielten Einsatz der Erdungsmaßnahmen. Das Erden der Steuerung dient als Schutz- und Funktionsmaßnahme.
  - Verbinden Sie Anlagenteile und Schränke mit dem System 200V sternförmig mit dem Erde/Schutzleitersystem. Sie vermeiden so die Bildung von Erdschleifen.
  - Verlegen Sie bei Potenzialdifferenzen zwischen Anlagenteilen und Schränken ausreichend dimensionierte Potenzialausgleichsleitungen.

Schirmung von Leitungen

Elektrische, magnetische oder elektromagnetische Störfelder werden durch eine Schirmung geschwächt; man spricht hier von einer Dämpfung.

Über die mit dem Gehäuse leitend verbundene Schirmschiene werden Störströme auf Kabelschirme zur Erde hin abgeleitet. Hierbei ist darauf zu achten, dass die Verbindung zum Schutzleiter impedanzarm ist, da sonst die Störströme selbst zur Störquelle werden.

Bei der Schirmung von Leitungen ist folgendes zu beachten:

- Verwenden Sie möglichst nur Leitungen mit Schirmgeflecht.
- Die Deckungsdichte des Schirmes sollte mehr als 80% betragen.
- In der Regel sollten Sie die Schirme von Leitungen immer beidseitig auflegen. Nur durch den beidseitigen Anschluss der Schirme erreichen Sie eine gute Störunterdrückung im höheren Frequenzbereich.

  Nur im Ausnahmefall kann der Schirm auch einseitig aufgelegt werden. Dann erreichen Sie jedoch nur eine Dämpfung der niedrigen Frequenzen. Eine einseitige Schirmanbindung kann günstiger sein, wenn:
  - die Verlegung einer Potenzialausgleichsleitung nicht durchgeführt werden kann
  - Analogsignale (einige mV bzw. µA) übertragen werden
  - Folienschirme (statische Schirme) verwendet werden.
- Benutzen Sie bei Datenleitungen für serielle Kopplungen immer metallische oder metallisierte Stecker. Befestigen Sie den Schirm der Datenleitung am Steckergehäuse. Schirm nicht auf den PIN 1 der Steckerleiste auflegen!
- Bei stationärem Betrieb ist es empfehlenswert, das geschirmte Kabel unterbrechungsfrei abzuisolieren und auf die Schirm-/Schutzleiterschiene aufzulegen.
- Benutzen Sie zur Befestigung der Schirmgeflechte Kabelschellen aus Metall. Die Schellen müssen den Schirm großflächig umschließen und guten Kontakt ausüben.
- Legen Sie den Schirm direkt nach Eintritt der Leitung in den Schrank auf eine Schirmschiene auf. Führen Sie den Schirm bis zum System 200V Modul weiter, legen Sie ihn dort jedoch **nicht** erneut auf!

Bitte bei der Montage beachten!

Bei Potenzialdifferenzen zwischen den Erdungspunkten kann über den beidseitig angeschlossenen Schirm ein Ausgleichsstrom fließen.

Abhilfe: Potenzialausgleichsleitung.

Allgemeine Daten

Aufbau/Maße

- Profilschiene 35mm
- Peripherie-Module mit seitlich versenkbaren Beschriftungsstreifen
- Maße Grundgehäuse:
  1fach breit: (HxBxT) in mm: 76x25,4x74 in Zoll: 3x1x3
  2fach breit: (HxBxT) in mm: 76x50,8x74 in Zoll: 3x2x3

Betriebssicherheit

- Anschluss über Federzugklemmen an Frontstecker, Aderquerschnitt 0,08 ... 2,5mm^2 bzw. 1,5mm^2 (18-fach Stecker)
- Vollisolierung der Verdrahtung bei Modulwechsel
- Potenzialtrennung aller Module zum Rückwandbus

Allgemeine Daten

Konformität und Approbation		
Konformität		
CE	2006/95/EG	Niederspannungsrichtlinie
	2004/108/EG	EMV-Richtlinie
Approbation		
UL	UL 508	Zulassung für USA und Kanada
Sonstiges		
RoHS	2011/65/EU	Produkte bleifrei; Richtlinie zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten

Personenschutz und Geräteschutz		
Schutzart	-	IP20
Potenzialtrennung		
Zum Feldbus	-	Galvanisch entkoppelt
Zur Prozessebene	-	Galvanisch entkoppelt
Isolationsfestigkeit	EN 61131-2	-
Isolationsspannung gegen Bezugserde		
Eingänge / Ausgänge	-	AC / DC 50V, bei Prüfspannung AC 500V
Schutzmaßnahmen	-	gegen Kurzschluss

Umgebungsbedingungen gemäß EN 61131-2		
Klimatisch		
Lagerung /Transport	EN 60068-2-14	-25…+70°C
Betrieb		
Horizontaler Einbau	EN 61131-2	0…+60°C
Vertikaler Einbau	EN 61131-2	0…+60°C
Luftfeuchtigkeit	EN 60068-2-30	RH1 (ohne Betauung, relative Feuchte 10 … 95%)
Verschmutzung	EN 61131-2	Verschmutzungsgrad 2
Mechanisch		
Schwingung	EN 60068-2-6	1g, 9Hz ... 150Hz
Schock	EN 60068-2-27	15g, 11ms

Montagebedingungen		
Einbauort	-	Im Schaltschrank
Einbaulage	-	Horizontal und vertikal

EMV	**Norm**		**Bemerkungen**
Störaussendung	EN 61000-6-4		Class A (Industriebereich)
Störfestigkeit Zone B	EN 61000-6-2		Industriebereich
		EN 61000-4-2	ESD 8kV bei Luftentladung (Schärfegrad 3), 4kV bei Kontaktentladung (Schärfegrad 2)
		EN 61000-4-3	HF-Einstrahlung (Gehäuse) 80MHz … 1000MHz, 10V/m, 80% AM (1kHz) 1,4GHz ... 2,0GHz, 3V/m, 80% AM (1kHz) 2GHz ... 2,7GHz, 1V/m, 80% AM (1kHz)
		EN 61000-4-6	HF-Leitungsgeführt 150kHz … 80MHz, 10V, 80% AM (1kHz)
		EN 61000-4-4	Burst, Schärfegrad 3
		EN 61000-4-5	Surge, Installationsklasse 3 *)

*) Aufgrund der energiereichen Einzelimpulse ist bei Surge eine angemessene externe Beschaltung mit Blitzschutzelementen wie z.B. Blitzstromableitern und Überspannungsableitern erforderlich.

Teil 2 Hardwarebeschreibung

Übersicht

In diesem Kapitel finden Sie Informationen über den Aufbau und die Anschlussbelegung des Kommunikationsprozessors CP 240 mit RS232-Schnittstelle.

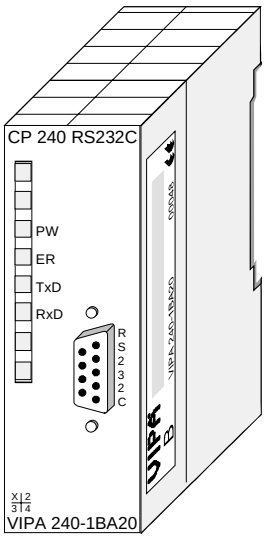
Inhalt

Thema	Seite
Teil 2 Hardwarebeschreibung	**2-1**
Leistungsmerkmale	2-2
Aufbau	2-3
Technische Daten	2-6

Leistungsmerkmale

CP 240 RS232
240-1BA20

- RS232-Schnittstelle
- Unterstützt werden die Protokolle ASCII, STX/ETX, 3964(R), RK512 und Modbus
- Parametrierung über 16Byte Parameterdaten
- Bis zu 250 Telegramme innerhalb der 1024Byte großen Empfangs- bzw. Sendepuffer
- Serielle Schnittstelle potenzialgetrennt zum Rückwandbus
- Spannungsversorgung über Rückwandbus

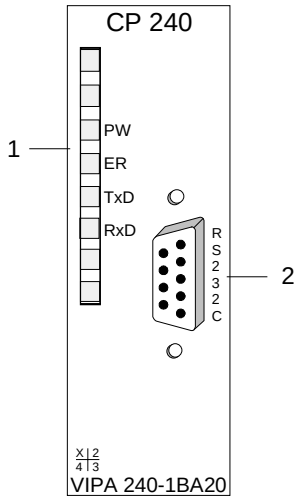


Bestelldaten

Typ	Bestellnummer	Beschreibung
CP 240 RS232	VIPA 240-1BA20	CP 240 mit RS232-Schnittstelle Protokolle: ASCII, STX/ETX, 3964(R), RK512, Modbus

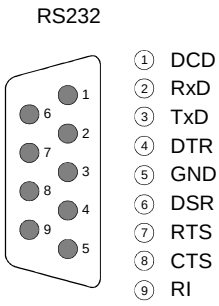
Aufbau

CP 240 RS232
240-1BA20



[1] LED Statusanzeigen
[2] 9poliger serieller SubD-Stecker für RS232-Kommunikation

Schnittstelle



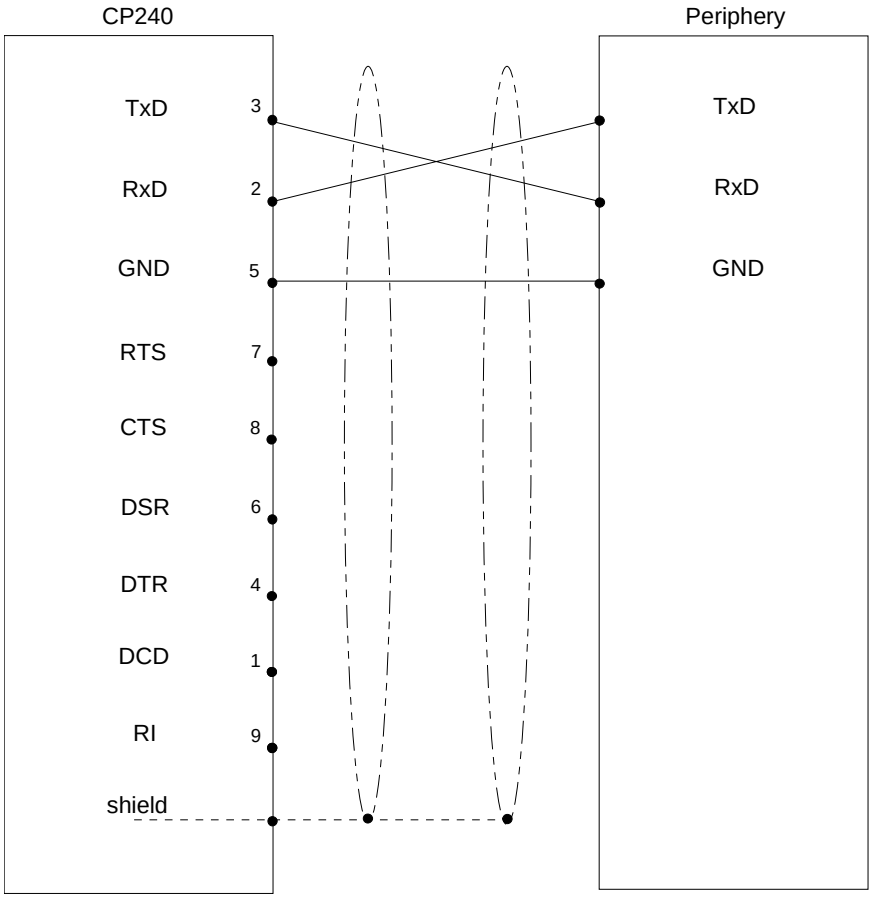
RS232-Schnittstelle

- Logische Zustände als Spannungspegel
- Punkt-zu-Punkt-Kopplung mit serieller Vollduplex-Übertragung
- Datenübertragung bis 15m Entfernung
- Datenübertragungsrate bis 115,2kBit/s

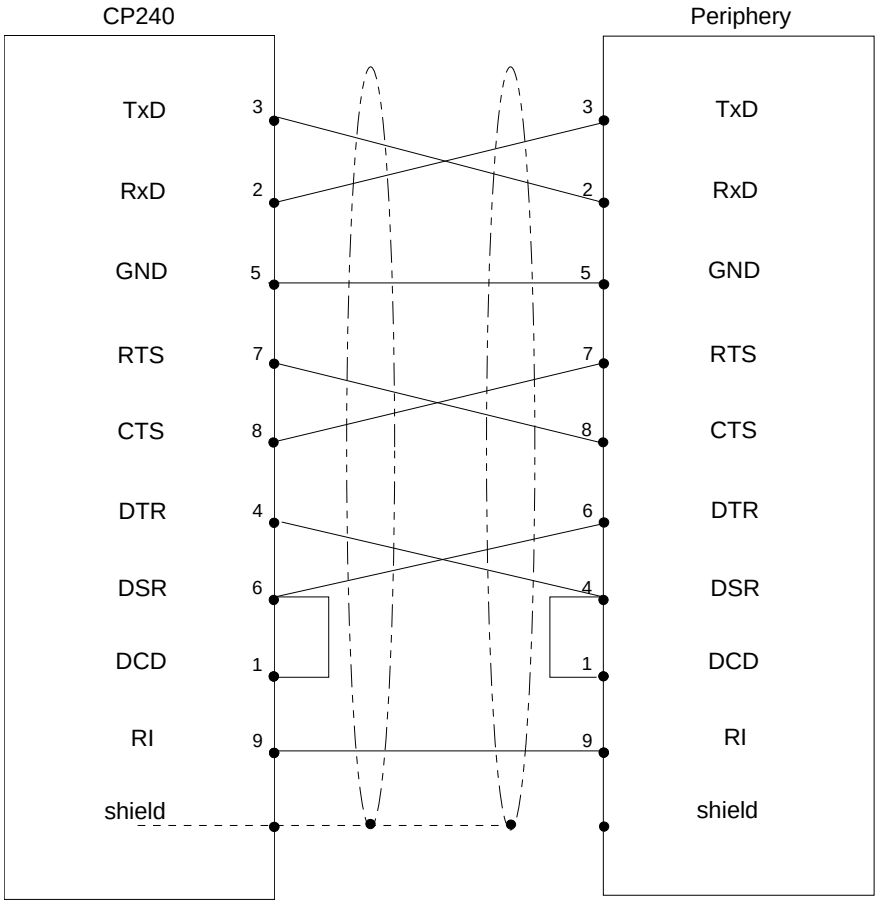
9poliger SubD-Stecker

Pin	Bezeichnung		Signalbeschreibung
1	DCD	Data Carrier Detect	Daten können empfangen werden
2	RxD	Receive Data	Empfangsdaten von Modem an CP 240
3	TxD	Transmit Data	Sendedaten von CP 240 an Modem
4	DTR	Data Terminal Ready	CP 240 ist betriebsbereit
5	GND	Signal Ground	GND Nullbezugspunkt
6	DSR	Data Set Ready	Modem signalisiert, dass es betriebsbereit ist
7	RTS	Request to send	CP 240 zeigt an, dass er betriebsbereit ist
8	CTS	Clear to send	Modem zeigt an, dass CP 240 senden darf
9	RI	Ring indicator	Klingelzeichen

RS232-Verkabelung ohne Hardware-Handshake



RS232-Verkabelung mit Hardware-Handshake



Spannungs-versorgung

Der Kommunikationsprozessor bezieht seine Versorgungsspannung über den Rückwandbus.

LEDs

Der Kommunikationsprozessor besitzt 4 LEDs, die der Betriebszustand-Anzeige dienen. Die Bedeutung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

Bez.	Farbe	Bedeutung
PW	Grün	Signalisiert eine anliegende Betriebsspannung
ER	Rot	Bei Modbus: Signalisiert internen Fehler. ansonsten: Signalisiert einen Fehler durch: Leitungsunterbrechung, Überlauf, Paritätsfehler oder Zeichenrahmenfehler. Automatisches Rücksetzen der Fehler-LED nach 4s. Bei aktivierter Diagnose erfolgt dann die Fehleranzeige über die Diagnosebytes.
TxD	Grün	Daten senden (transmit data)
RxD	Grün	Daten empfangen (receive data)

Technische Daten

Artikelnr.	**240-1BA20**
Bezeichnung	CP 240, PtP RS232
Stromaufnahme/Verlustleistung	
Stromaufnahme aus Rückwandbus	150 mA
Verlustleistung	0,75 W
Status, Alarm, Diagnosen	
Statusanzeige	ja
Alarme	nein
Prozessalarm	nein
Diagnosealarm	nein
Diagnosefunktion	nein
Diagnoseinformation auslesbar	möglich
Versorgungsspannungsanzeige	ja
Sammelfehleranzeige	rote LED
Kanalfehleranzeige	keine
Funktionalität Sub-D Schnittstellen	
Bezeichnung	-
Physik	RS232
Anschluss	9poliger SubD Stecker
Potenzialgetrennt	✓
MPI	-
MP²I (MPI/RS232)	-
Punkt-zu-Punkt-Kopplung	✓
Point-to-Point Kommunikation	
PtP-Kommunikation	✓
Schnittstelle potentialgetrennt	✓
Schnittstelle RS232	✓
Schnittstelle RS422	-
Schnittstelle RS485	-
Anschluss	9poliger SubD Stecker
Übertragungsgeschwindigkeit, min.	150 bit/s
Übertragungsgeschwindigkeit, max.	115,2 kbit/s
Leitungslänge, max.	15 m
Point-to-Point Protokolle	
Protokoll ASCII	✓
Protokoll STX/ETX	✓
Protokoll 3964(R)	✓
Protokoll RK512	✓
Protokoll USS Master	-
Protokoll Modbus Master	✓
Protokoll Modbus Slave	✓
Spezielle Protokolle	-
Datengrößen	
Eingangsbytes	16
Ausgangsbytes	16
Parameterbytes	16
Diagnosebytes	0
Gehäuse	
Material	PPE
Befestigung	Profilschiene 35mm
Mechanische Daten	
Abmessungen (BxHxT)	25,4 x 76 x 78 mm
Gewicht	80 g

Artikelnr.	**240-1BA20**
Umgebungsbedingungen	
Betriebstemperatur	0 °C bis 60 °C
Lagertemperatur	-25 °C bis 70 °C
Zertifizierungen	
Zertifizierung nach UL508	ja

Teil 3 Einsatz

Übersicht

Den Kommunikationsprozessor CP 240 erhalten Sie von VIPA mit verschiedenen Übertragungsprotokollen, auf deren Einsatz hier näher eingegangen wird.

Inhalt

Thema	Seite
Teil 3 Einsatz	**3-1**
Schnelleinstieg	3-2
GSD und FCs einbinden	3-4
Projektierung	3-5
Standardhantierungsbausteine	3-8
RK512-Kommunikation - Hantierungsbausteine	3-13
RK512-Kommunikation - Anzeigewort ANZW	3-18
ASCII / STX/ETX / 3964(R) / RK512 - Grundlagen	3-20
ASCII / STX/ETX / 3964(R) / RK512 - Kommunikationsprinzip	3-26
ASCII / STX/ETX / 3964(R) / RK512 - Parametrierung	3-29
Modbus - Grundlagen	3-36
Modbus - Parametrierung	3-38
Modbus - Einsatz	3-41
Modbus - Funktionscodes	3-45
Modbus - Fehlermeldungen	3-49
Modbus - Beispiel	3-50

Schnelleinstieg

Übersicht

Die Adresszuordnung und die Parametrierung des CP 240 erfolgt im Siemens SIMATIC Manager in Form eines virtuellen PROFIBUS-Systems. Hierzu ist die Einbindung der VIPA_21x.gsd (ab V. 1.67) erforderlich.

Für die Kommunikation zwischen Ihrer CPU und dem CP 240 sind Hantierungsbausteine in Form einer Bibliothek verfügbar, die Sie in Ihren Siemens SIMATIC Manager einbinden können.

Vorgehensweise

Vorbereitung

- Starten Sie den Siemens SIMATIC Manager mit einem neuen Projekt.
- Binden Sie die VIPA_21x.gsd ein. Verwenden Sie hierbei eine GSD-Version ab V. 1.67.
- Binden Sie die Bausteinbibliothek ein, indem Sie die Vipa_Bibliothek_*Vxxx.zip* entpacken und die Datei VIPA.ZIP dearchivieren.
- Öffnen Sie die Bibliothek und übertragen Sie die gewünschten FCs in Ihr Projekt.

Hardware-Konfiguration

Für die Hardwarekonfiguration verfahren Sie auf die gleiche Weise wie im Handbuch HB97 - CPU beschrieben:

- Projektieren Sie ein PROFIBUS-DP-Mastersystem mit der Siemens CPU 315-2DP (6ES7 315-2AF03 V1.2) und legen Sie ein PROFIBUS-Subnetz an.
- Binden Sie an das Master-System aus dem Hardware-Katalog das Slave-System "VIPA_CPU21x" an. Sie finden das Slave-System im Hardware-Katalog unter
  PROFIBUS-DP > Weitere Feldgeräte > I/O > VIPA_System_200V.
- Geben Sie dem Slave-System die Adresse 1. Hiermit identifiziert die VIPA CPU das System als zentrales Peripherie-System.
- Platzieren Sie in diesem Slave-System in der gesteckten Reihenfolge Ihre Module. Beginnen sie mit der CPU auf dem 1. Steckplatz.
- Binden Sie danach Ihre System 200V Module und an der entsprechenden Stelle Ihren CP 240 ein.
- Parametrieren Sie ggf. Ihren CP 240.

Parameter

Zur Parametrierung können dem CP 16Byte Parameterdaten übergeben werden, die je nach gewähltem Protokoll entsprechend belegt sind.

Die Parametrierung erfolgt über die Hardware-Konfiguration im Siemens SIMATIC Manager durch Einbindung eines protokollspezifischen CP 240.

Protokolle

Nach der GSD-Einbindung ist der CP 240 mit folgenden Protokollen verfügbar:

- ASCII
- STX/ETX
- 3964(R) und RK512
- Modbus (Master, Slave)

Kommunikation

Die serielle Kommunikation erfolgt unter Einsatz von Hantierungsbausteinen im SPS-Anwenderprogramm. Die Hantierungsbausteine finden Sie auf www.vipa.com im Service-Bereich.

Zur internen Kommunikation sind VIPA FCs zu verwenden. Hier werden Daten mit einer maximalen Blockgröße von 12Byte übertragen.

Je nach Protokoll kommen folgende Hantierungsbausteine zum Einsatz:

ASCII	STX/ETX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x	x	FC8	STEUERBIT
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

Hinweis!

Außer bei Modbus ist eine Kommunikation mit SEND- und RECEIVE-Bausteinen nur möglich, wenn zuvor im Anlauf-OB der Parameter ANL des SYNCHRON-Bausteins gesetzt wurde.

GSD und FCs einbinden

Projektierung über GSD

Adresszuordnung und die Parametrierung des CP 240 erfolgt im Siemens SIMATIC Manager in Form eines virtuellen PROFIBUS-Systems. Da die PROFIBUS-Schnittstelle softwareseitig standardisiert ist, können wir auf diesem Weg gewährleisten, dass über die Einbindung einer GSD-Datei die Funktionalität in Verbindung mit dem SIMATIC Manager von Siemens jederzeit gegeben ist. Ihr Projekt übertragen Sie über MPI in die CPU.

GSD einbinden

Folgende Schritte sind zur Installation der GSD erforderlich:

- Im Service-Bereich von www.vipa.com finden Sie die GSD-Datei für das System 200V. Laden Sie die zip-Datei auf Ihren PC.
- Starten Sie mit einem Doppelklick auf die Datei Ihr Unzip-Programm und entpacken Sie die Daten in Ihr Arbeitsverzeichnis.
- Kopieren Sie die GSD-Datei **`VIPA_21X.GSD`** in Ihr GSD-Verzeichnis ... \siemens\step7\s7data\gsd
- Starten Sie den Hardware-Konfigurator von Siemens
- Schließen Sie alle Projekte
- Gehen Sie auf **Extras** > *Neue GSD-Datei installieren*
- Geben Sie hier **VIPA_21X.gsd** an

Die Module des System 200V von VIPA sind jetzt im Hardwarekatalog integriert und können projektiert werden.

Bausteine installieren

Die VIPA-spezifischen Bausteine finden Sie im Service-Bereich auf www.vipa.com als Bibliothek zum Download. Die Bibliothek liegt als gepackte zip-Datei vor.

Sobald Sie VIPA-spezifische Bausteine verwenden möchten, sind diese in Ihr Projekt zu importieren.

Bibliothek dearchivieren

Starten Sie mit einem Doppelklick auf die Datei Vipa_Bibliothek_Vxxx.zip Ihr Unzip-Programm und kopieren Sie die Datei vipa.zip in Ihr Arbeitsverzeichnis. Es ist nicht erforderlich diese Datei weiter zu entpacken.

Zur Dearchivierung Ihrer Bibliothek für die SPEED7-CPUs starten Sie den SIMATIC Manager von Siemens. Über **Datei** > *Dearchivieren* öffnen Sie ein Dialogfenster zur Auswahl des Archivs. Navigieren Sie in Ihr Arbeitsverzeichnis.

Wählen Sie VIPA.ZIP an und klicken Sie auf [Öffnen].

Geben Sie ein Zielverzeichnis an, in dem die Bausteine abzulegen sind. Mit [OK] startet der Entpackvorgang.

Bibliothek öffnen und Bausteine in Projekt übertragen

Öffnen Sie die Bibliothek nach dem Entpackvorgang.

Öffnen Sie Ihr Projekt und kopieren Sie die erforderlichen Bausteine aus der Bibliothek in das Verzeichnis "Bausteine" Ihres Projekts.

Nun haben Sie in Ihrem Anwenderprogramm Zugriff auf die VIPA-spezifischen Bausteine.

Projektierung

Allgemein

Die Adresszuordnung und die Parametrierung der direkt gesteckten System 200V Module erfolgt im SIMATIC Manager von Siemens in Form eines virtuellen PROFIBUS-Systems. Ihr Projekt übertragen Sie seriell über die MPI-Schnittstelle oder über MMC in Ihre CPU.

Voraussetzung

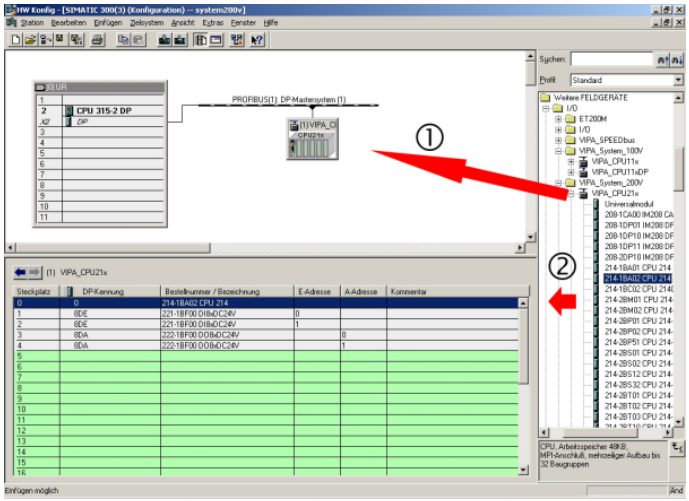
Für die Projektierung der CPU werden fundierte Kenntnisse im Umgang mit dem SIMATIC Manager und dem Hardware-Konfigurator von Siemens vorausgesetzt!

Folgende Voraussetzungen müssen für die Projektierung erfüllt sein:

- SIMATIC Manager von Siemens auf PC bzw. PG installiert
- GSD-Dateien in Hardware-Konfigurator von Siemens eingebunden
- Projekt kann in CPU übertragen werden (seriell z.B. "Green Cable" oder MMC)

Hardware-Konfiguration

- Starten Sie den Hardware-Konfigurator von Siemens mit einem neuen Projekt und fügen Sie aus dem Hardware-Katalog eine Profilschiene ein.
- Platzieren Sie auf dem ersten möglichen Steckplatz die CPU 315-2DP (6ES7 315-2AF03 V1.2) von Siemens.
- Sofern Ihre CPU 21x einen PROFIBUS-DP-Master integriert hat, können Sie diesen jetzt mit PROFIBUS vernetzen und Ihre DP-Slaves anbinden.
- Erzeugen Sie ein PROFIBUS-Subnetz (falls noch nicht vorhanden).
- Hängen Sie an das Subnetz das System "VIPA_CPU21x". Sie finden dies im Hardware-Katalog unter *PROFIBUS DP* > *Weitere Feldgeräte* > *IO* > VIPA_System_200V. Geben Sie diesem Slave die **PROFIBUS-Adresse 1**.
- Platzieren Sie in Ihrem Konfigurator **immer auf dem 1. Steckplatz** die CPU 21x, die Sie einsetzen, indem Sie diese dem Hardware-Katalog entnehmen.
- Binden Sie danach Ihre System 200V Module in der gesteckten Reihenfolge und an der entsprechenden Stelle Ihren CP 240 ein.
- Parametrieren Sie ggf. Ihren CP 240.
- Sichern Sie Ihr Projekt.



SPS-Programm

Für die nachfolgend gezeigte Kommunikation zwischen CPU und CP 240 kommen folgende Hantierungsbausteine zum Einsatz:

FC 0	SEND	Datenausgabe CPU an CP 240
FC 1	RECEIVE	Datenempfang vom CP 240
FC 9	SYNCHRON_RESET	Synchronisation zwischen CPU und CP 240

Die Hantierungsbausteine sind als Bibliothek verfügbar und können, wie weiter oben gezeigt, im Siemens SIMATIC Manager eingebunden werden.

Eine nähere Beschreibung der Hantierungsbausteine finden Sie auf den Folgeseiten. Ihr SPS-Programm sollte nach folgender Struktur aufgebaut sein:

```
OB1:
      CALL  FC     9             //Synchron aufrufen
       ADR       :=0             //1. DW im SEND/EMPF_DB
       TIMER_NR  :=T2            //Wartezeit Synchron
       ANL       :=M3.0          //Anlauf erfolgt
       NULL      :=M3.1          //Zwischenmerker
       RESET     :=M3.2          //Baugruppenreset ausführen
       STEUERB_S :=MB2           //Steuerbits Sende_FC
       STEUERB_R :=MB1           //Steuerbits Receive_FC
      U     M      3.0           //solange Anlauf keine
                                 //SEND/RECEIVE Bearbeitung
      BEB

      CALL  FC     1             //Receive Daten
       ADR          :=0          //1. DW im SEND/EMPF_DB
       _DB          :=DB11       //Empfang_DB Telegramm
       ABD          :=W#16#14    //1. DW Empfangspuffer (DW20)
       ANZ          :=MW10       //Anzahl empfangener Daten
       EMFR         :=M1.0       //Empfang fertig
       PAFE         :=MB12       //Fehlerbyte
       GEEM         :=MW100      //Interne Daten
       ANZ_INT      :=MW102      //Interne Daten
       empf_laeuft  :=M1.1       //Interne Daten
       letzter_block:=M1.2       //Interne Daten
       fehl_empf    :=M1.3       //Interne Daten
      U     M      1.0           //Empfang fertig
      R     M      1.0           //loesche Empfang fertig
      CALL  FC     0             //Sende Daten
       ADR          :=0          //1. DW im SEND/EMPF_DB
       _DB          :=DB10       //Sende_DB Telegramm
       ABD          :=W#16#14    //1. DW Sendepuffer (DW20)
       ANZ          :=MW14       //Anzahl zu sendender Daten
       FRG          :=M2.0       //Senden fertig angeben
       PAFE         :=MB16       //Fehlerbyte
       GESE         :=MW104      //Interne Daten
       ANZ_INT      :=MW106      //Interne Daten
       ende_kom     :=M2.1       //Interne Daten
       letzter_block:=M2.2       //Interne Daten
       senden_laeuft:=M2.3       //Interne Daten
       fehler_kom   :=M2.4       //Interne Daten


OB100:
      UN    M      3.0
      S     M      3.0           //Anlauf der CPU erfolgt
```

Projekt übertragen

Die Datenübertragung erfolgt über MPI. Sollte Ihr Programmiergerät keine MPI-Schnittstelle besitzen, können Sie für eine serielle Punkt-zu-Punkt-Übertragung von Ihrem PC an MPI das "Green Cable" von VIPA verwenden.

Das "Green Cable" hat die Best.-Nr. VIPA 950-0KB00 und darf nur bei den VIPA CPUs mit MP^{2}I-Schnittstelle eingesetzt werden.

Bitte beachten Sie hierzu die Hinweise zum Green Cable in den Grundlagen!

- Verbinden Sie Ihr PG mit der CPU.
- Mit **Zielsystem** > *Laden in Baugruppe* in Ihrem Projektiertool übertragen Sie Ihr Projekt in die CPU.
- Stecken Sie eine MMC und übertragen Sie mit **Zielsystem** > *RAM nach ROM kopieren* Ihr Anwenderprogramm auf die MMC.
- Während des Schreibvorgangs blinkt die "MC"-LED auf der CPU. Systembedingt wird zu früh ein erfolgter Schreibvorgang gemeldet. Der Schreibvorgang ist erst beendet, wenn die LED erlischt.

Was ist das Green Cable ?

Das Green Cable ist ein grünes Verbindungskabel, das ausschließlich zum Einsatz an VIPA System-Komponenten konfektioniert ist.

Mit dem Green Cable können Sie:

- Projekte Punkt-zu-Punkt seriell übertragen
- Firmware-Updates der CPUs und Feldbus-Master durchführen

Wichtige Hinweise zum Einsatz des Green Cable

Bei Nichtbeachtung der nachfolgenden Hinweise können Schäden an den System-Komponenten entstehen.

Für Schäden, die aufgrund der Nichtbeachtung dieser Hinweise und bei unsachgemäßem Einsatz entstehen, übernimmt die VIPA keinerlei Haftung!

Hinweis zum Einsatzbereich

Das Green Cable darf ausschließlich direkt an den hierfür vorgesehenen Buchsen der VIPA-Komponenten betrieben werden (Zwischenstecker sind nicht zulässig). Beispielsweise ist vor dem Stecken des Green Cable ein gestecktes MPI-Kabel zu entfernen.

Zurzeit unterstützen folgende Komponenten das Green Cable:

VIPA CPUs mit MP^{2}I-Buchse sowie die Feldbus-Master von VIPA.

Hinweis zur Verlängerung

Die Verlängerung des Green Cable mit einem weiteren Green Cable bzw. die Kombination mit weiteren MPI-Kabeln ist nicht zulässig und führt zur Beschädigung der angeschlossenen Komponenten!

Das Green Cable darf nur mit einem 1:1 Kabel (alle 9 Pin 1:1 verbunden) verlängert werden.

Standardhantierungsbausteine

SEND (FC 0)

Dieser FC dient zur Datenausgabe von der CPU an den CP 240. Hierbei legen Sie über die Bezeichner _DB, ADB und ANZ den Sendebereich fest.

Über das Bit FRG wird der Sendeanstoß gesetzt und die Daten werden gesendet. Nach dem Übertragen der Daten setzt der Hantierungsbaustein das Bit FRG wieder zurück.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing data to send
in	ABD	WORD	No. of 1. data word to send
in	ANZ	WORD	No of bytes to send
in_out	FRG	BOOL	Start bit of the function
in_out	GESE	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	ENDE_KOMM	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	Status of function
in_out	FEHLER_KOM	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbausteins, der die zu sendenden Daten beinhaltet.

ABD Wortvariable, welche die Nummer des Datenworts enthält, ab dem die auszugebenden Zeichen abgelegt sind.

ANZ Anzahl der Bytes, die zu übertragen sind.

FRG Sendefreigabe Bei FRG = "1" werden die über _DB, ADB und ANZ definieren Daten einmalig an den über ADR adressierten CP übertragen. Nach der Übertragung wird FRG wieder zurückgesetzt. Ist beim Aufruf FRG = "0", wird der Baustein sofort wieder verlassen!

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GESE, ANZ_INT ENDE_KOM LETZTER_BLOCK SENDEN_LAEUFT FEHLER_KOM Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits ENDE_KOM, LETZTER _BLOCK, SENDEN_LAEUFT und FEHLER_KOM immer in einem Merker-Byte abzulegen.

RECEIVE (FC 1)

Dieser FC dient zum Datenempfang vom CP 240. Hierbei legen Sie über die Bezeichner _DB und ADB den Empfangsbereich fest.

Ist der Ausgang EMFR gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in ANZ abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1. data word received
out	ANZ	WORD	No of bytes received
out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	Status of function
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)
in_out	OFFSET	WORD	internal use

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbaustein, der die empfangenen Daten beinhaltet.

ABD Wortvariable, welche die Nummer des Datenworts enthält, ab dem die empfangenen Zeichen abgelegt sind.

ANZ Wortvariable, welche die Anzahl der Bytes enthält, die empfangen wurden.

EMFR Durch Setzen des EMFR zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von EMFR im Anwenderprogramm können weitere Daten empfangen werden.

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF
OFFSET

Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

STEUERBIT (FC 8) Mit diesem Baustein haben Sie folgenden Zugriff auf die seriellen Modemleitungen:

Lesen: DTR, RTS, DSR, RI, CTS, CD

Schreiben: DTR, RTS

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	RTS	BOOL	New state RTS
in	DTR	BOOL	New state DTR
in	MASKE_RTS	BOOL	0: do nothing 1: set state RTS
in	MASKE_DTR	BOOL	0: do nothing 1: set state DTR
out	STATUS	BYTE	Status flags
out	DELTA_STATUS	BYTE	Status flags of change between 2 accesses
in_out	START	BOOL	Start bit of the function
in_out	AUFTRAG_LAEU	BOOL	Status of function
out	RET_VAL	WORD	Return Code (00=OK)

Hinweis!

Dieser Baustein darf nicht aufgerufen werden, solange ein Sendeauftrag läuft, ansonsten kann dies zu Datenverlust führen.

ADR Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

RTS, DTR Mit diesem Parameter geben Sie den Status für RTS bzw. DTR vor, den Sie über MASK_RTS bzw. MASK_DTR aktivieren können.

MASK_RTS, MASK_DTR Hier wird mit 1 der Status des entsprechenden Parameters übernommen, sobald Sie START auf 1 setzen.

STATUS, DELTA_STATUS STATUS liefert den aktuellen Status der Modem-Leitungen zurück. DELTA_STATUS liefert den Status der Modem-Leitungen zurück, die sich seit dem letzten Zugriff geändert haben.

Die Bytes haben folgenden Aufbau:

Bit-Nr.	7	6	5	4	3	2	1	0
STATUS	x	x	RTS	DTR	CD	RI	DSR	CTS
DELTA_STATUS	x	x	x	x	CD	RI	DSR	CTS

START Durch Setzen von START wird der über die Maske aktivierte Status übernommen.

AUFTRAG_LAEU Solange die Funktion abgearbeitet wird, bleibt dieses Bit gesetzt.

RET_VAL Dieser Parameter liefert zur Zeit immer 00h zurück und dient zukünftigen Fehlermeldungen.

SYNCHRON_ RESET
Synchronisation und Rücksetzen (FC 9)

Der Baustein ist im zyklischen Programmteil aufzurufen. Mit dieser Funktion wird die Anlaufkennung des CP 240 quittiert, und so die Synchronisation zwischen CPU und CP hergestellt. Weiterhin kann bei einer Kommunikationsunterbrechung der CP rückgesetzt werden und so ein synchroner Anlauf erfolgen.

Hinweis!

Außer bei Modbus ist eine Kommunikation mit SEND- und RECEIVE-Bausteinen nur möglich, wenn zuvor im Anlauf-OB der Parameter ANL des SYNCHRON-Bausteins gesetzt wurde.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	TIMER_NR	WORD	No of timer for idle time
in_out	ANL	BOOL	restart progressed
in_out	NULL	BOOL	internal use
in_out	RESET	BOOL	1 = Reset the CP
in_out	STEUERB_S	BYTE	internal use
in_out	STEUERB_R	BYTE	internal use

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

TIMER_NR Nummer des Timers für die Wartezeit.

ANL Mit ANL = 1 wird dem Hantierungsbaustein mitgeteilt, dass an der CPU STOP/START bzw. NETZ-AUS/NETZ-EIN erfolgt ist und nun eine Synchronisation erfolgen muss. Nach der Synchronisation wird ANL automatisch zurückgesetzt.

NULL Parameter wird intern verwendet.

RESET Mit RESET = 1 können Sie den CP aus Ihrem Anwenderprogramm zurücksetzen.

STEUERB_S Hier ist das Merkerbyte anzugeben, in dem die Steuerbits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT und FEHLER_KOM für den SEND-FC abgelegt sind.

STEUERB_R Hier ist das Merkerbyte anzugeben, in dem die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF für den RECEIVE-FC abgelegt sind.

ASCII_FRAGMENT (FC 11)

Dieser FC dient zum fragmentierten ASCII-Datenempfang. Hiermit haben Sie die Möglichkeit große Telegramme in 12Byte-Blöcken direkt nach dem Erhalt an die CPU weiterzureichen. Hierbei wartet der CP nicht, bis das komplette Telegramm empfangen wurde. Der Einsatz des FC 11 setzt voraus, dass Sie beim Empfänger "ASCII-fragmentiert" parametriert haben.

Im FC 11 legen Sie über die Bezeichner _DB und ADB den Empfangsbereich fest. Ist der Ausgang EMFR gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in ANZ abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1. data word received
out	ANZ	WORD	No of bytes received
in_out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbaustein, der die empfangenen Daten beinhaltet.

ABD Wortvariable, welche die Nummer des Datenworts enthält, ab dem die empfangenen Zeichen abgelegt sind.

ANZ Wortvariable, welche die Anzahl der Bytes enthält, die empfangen wurden.

EMFR Durch Setzen des EMFR zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von EMFR im Anwenderprogramm können weitere Daten empfangen werden.

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GEEM, ANZ_INT LETZTER_BLOCK EMPF_LAEUFT FEHLER_EMPF Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_REST (FC9) sind die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

RK512-Kommunikation - Hantierungsbausteine

FETCH_RK512 (FC 2)

Dieser FC dient dem aktiven Zugriff mittels RK512 auf eine Partner-Station, welche passiv Daten zur Verfügung stellt. Hierbei wird ein Telegramm mit den Quelldaten an die Partner-Station gesendet. Die Partner-Station stellt die Daten zusammen und sendet diese zurück.

Die empfangenen Daten werden im Ziel-DB abgelegt.

Hierbei legen Sie über die Bezeichner QDB, QBDW und LANG den Quellbereich in der Partner-Station und mit ZDB und ZDBW den Zielbereich in der eigenen Station fest.

Beim Aufruf des FCs wird anhand der Steuerbits geprüft, ob noch ein laufender Auftrag vorhanden ist. Sind alle Steuerbits null, so wird ein neuer FETCH-Auftrag angestoßen. Hierzu wird der Telegrammkopf an den CP übergeben und anschließend auf die Quittung mit den Nutzdaten gewartet.

Solange das Quittungstelegramm mit den Nutzdaten nicht gesendet wurde, ist im Anzeigewort "Auftrag läuft" gesetzt. Erst nachdem der Empfang des Quittungstelegramms vom CP an die SPS gemeldet und die Nutzdaten übergeben wurden, wird "Auftrag fertig" im Anzeigewort gesetzt und die Kommunikation mit dem CP beendet.

Die Funktion ist so lange im zyklischen Programm zu bearbeiten bis "Auftrag fertig mit/ohne Fehler" im Anzeigewort gesetzt ist.

Bei einer fehlerhaften Kommunikation übergibt der CP eine Fehlernummer an die SPS. Daraufhin wird im Anzeigewort die Fehlernummer eingetragen und das Bit "Auftrag fertig mit Fehler" gesetzt. Anschließend wird die Kommunikation mit dem CP beendet.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	QDB	BLOCK_DB	DB No. of DB of the remote station
in	QBDW	WORD	No. of 1. DW of the DB of remote station
in	LANG	INT	Length of data to transfer
in	ZDB	BLOCK_DB	Number target DB of this station
in	ZDBW	INT	No. of 1. data word in target DB
in	KOOR	WORD	Coordination flag
out	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	ANZ	WORD	internal use
in_out	GESE	WORD	internal use
in_out	KOPF_GESE	BOOL	internal use
in_out	WART_DATEN	BOOL	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHL_KOM	BOOL	internal use

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

QDB Nummer des Quell-Datenbausteins in der Partner-Station.

QBDW 1. Datenwort im Quell-Datenbaustein in der Partner-Station

LANG Länge der zu sendenden Daten in Worten

ZDB Nummer des Ziel-Datenbausteins in der eigenen Station

ZDBW 1. Datenwort im Ziel-Datenbaustein

KOOR Koordinierungsmerker

Der Koordinierungsmerker dient zum koordinierten Empfangen von Daten. Mit einem FETCH-Auftrag wird der Koordinierungsmerker gesetzt. Solange der Merker gesetzt ist, kann kein weiterer FETCH-Auftrag angestoßen werden. Der Einsatz eines Koordinierungsmerkers ist dann sinnvoll, wenn Sie verhindern möchten, dass Daten nach dem Empfang durch neue überschrieben werden.

Durch Angabe von FFFFh ist der Koordinierungsmerker deaktiviert.

ANZW Anzeigewort

Über das Anzeigewort können Sie Informationen zur Auftragsbearbeitung abrufen. Näheres hierzu finden Sie weiter unter "RK515-Kommunikation - Anzeigewort ANZW"

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

ANZ, GESE, KOPF_GESE, WART_DATEN, EMPF_LAEUFT, LETZTER_BLOCK, FEHL_KOM Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen:

SEND_RK512 (FC 3)

Dieser FC dient zur Datenausgabe von der CPU an eine Partner-Station. Mit den Daten wird das Ziel für die Partner-Station mitgeliefert.

Hierbei legen Sie über die Bezeichner QDB, QBDW und LANG den Quellbereich in der eigenen Station und mit ZDB und ZBDW den Zielbereich in der Partner-Station fest.

Beim Aufruf des FCs wird anhand der Steuerbits geprüft ob noch ein laufender Auftrag vorhanden ist. Sind alle Steuerbits null, so wird ein neuer Sendeauftrag angestoßen. Hierzu wird das Telegramm bestehend aus Telegrammkopf und Nutzdaten an den CP übergeben und anschließend auf die Quittung gewartet.

Solange das Quittungstelegramm vom Partner nicht gesendet wurde, ist im Anzeigewort "Auftrag läuft" gesetzt. Erst nachdem der Empfang des Quittungstelegramms vom CP an die SPS gemeldet wurde, wird "Auftrag fertig" im Anzeigewort gesetzt und die Kommunikation mit dem CP beendet.

Bei einer fehlerhaften Kommunikation übergibt der CP eine Fehlernummer an die SPS. Daraufhin wird im Anzeigewort die Fehlernummer eingetragen und das Bit "Auftrag fertig mit Fehler" gesetzt. Anschließend wird die Kommunikation mit dem CP beendet. Die Funktion ist so langer zyklisch im Programm zu bearbeiten bis "Auftrag fertig mit/ohne Fehler" im Anzeigewort gesetzt ist.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	QDB	BLOCK_DB	DB No. of DB of this station
in	QBDW	WORD	No. of 1. DW of the DB of this station
in	LANG	INT	Length of data to send
in	ZDB	BLOCK_DB	DB No. of DB of the remote station
in	ZDBW	INT	No of 1. DW of the DB of remote station
in	KOOR	WORD	Coordination flag
out	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	ANZ	WORD	internal use
in_out	GESE	WORD	internal use
in_out	KOPF_GESENDET	BOOL	internal use
in_out	ERSTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	internal use
in_out	SENDEN_FERTIG	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER	BOOL	internal use

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

QDB Nummer des Quell-Datenbausteins in der eigenen Station.

QBDW 1. Datenwort im Quell-Datenbaustein in der eigenen Station.

LANG Länge der zu sendenden Daten in Worten

ZDB	Nummer des Ziel-Datenbausteins in der Partner-Station
ZDBW	1. Datenwort im Ziel-Datenbaustein
KOOR	Der Koordinierungsmerker dient zum koordinierten Senden von Daten. Mit einem SEND-Auftrag wird der Koordinierungsmerker gesetzt. Solange der Merker gesetzt ist, kann kein weiterer SEND-Auftrag angestoßen werden. Durch Angabe von FFFFh ist der Koordinierungsmerker deaktiviert.
ANZW	Anzeigewort Über das Anzeigewort können Sie Informationen zur Auftragsbearbeitung abrufen. Näheres hierzu finden Sie weiter unter "RK515-Kommunikation - Anzeigewort ANZW"
PAFE	Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich: 1 = Datenbaustein nicht vorhanden 2 = Datenbaustein zu kurz 3 = Datenbausteinnummer nicht im gültigen Bereich
ANZ, GESE, KOPF_GESE, ERSTER_BLOCK, SENDEN_LAEUFT, SENDEN_FERTIG, LETZTER_BLOCK, FEHLER	Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen.

S/R_ALL_RK512 (FC 4)

Dieser FC dient dazu FETCH- und SEND-Aufträge in der passiven Station zu bearbeiten.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	GESE	WORD	internal use
in_out	ANZ	WORD	internal use
in_out	DB_KOPF	WORD	internal use
in_out	ABF_KOPF	WORD	internal use
in_out	KOPF_AUSGEW	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	ENDE_KOM	BOOL	internal use
in_out	SEND_ALL	BOOL	internal use
in_out	RECEIV_ALL	BOOL	internal use
in_out	FEHLER	BOOL	internal use

ADR

Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

ANZW

Anzeigewort

Über das Anzeigewort können Sie Informationen zur Auftragsbearbeitung abrufen. Näheres hierzu finden Sie weiter unter "RK515-Kommunikation - Anzeigewort ANZW"

PAFE

Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GESE, ANZ, DB_KOPF, ABF_KOPF, KOPF_AUSGEW, LETZTER_BLOCK, SENDE_LAEUFT, EMPF_LAEUFT, ENDE_KOM, SEND_ALL, RECEIVE_ALL, FEHLER

Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen.

RK512-Kommunikation - Anzeigewort ANZW

Status- und Fehleranzeigen

Status und Fehleranzeigen liefern die Hantierungsbausteine:

- über das Anzeigewort ANZW (Informationen zur Auftragsbearbeitung).
- über das Parametrierfehlerbyte PAFE (Anzeige einer fehlerhaften Auftragsparametrierung).

Inhalt und Aufbau Anzeigewort ANZW

Das "Anzeigewort" zeigt den Zustand für einen bestimmten Auftrag auf einem CP an.

Im SPS-Programm sollte für jeden Auftrag ein eigenes "Anzeigewort" für jeden definierten Auftrag bereitgestellt werden.

Das Anzeigewort hat den folgenden prinzipiellen Aufbau:

Byte	Bit 7 ... Bit 0
0	*Fehlermeldung CP* 00h: kein Fehler 17h: Telegramm zu lang 0Ch: Zeichenrahmen-Fehler 07h: Quittungsverzug 0Ah: DBL überschritten
1	*Statusverwaltung CPU* Bit 0: nicht belegt Bit 1: Auftrag läuft 0: SEND/FETCH freigegeben 1: SEND/FETCH gesperrt Bit 2: Auftrag fertig ohne Fehler Bit 3: Auftrag fertig mit Fehler Bit 7 ... Bit 4: nicht belegt

Fehlermeldung CP Byte 0

In diesem Byte wird die Fehlermeldungen eingetragen, welche der CP liefert. Die Fehlermeldungen sind nur gültig, wenn auch gleichzeitig das Bit "Auftrag fertig mit Fehler" in der *Statusverwaltung* gesetzt ist.

Folgende Fehlermeldungen können ausgegeben werden:

00h *kein Fehler*

Sollte das Bit Auftrag fertig mit Fehler gesetzt sein, hat der CP die Verbindung neu aufbauen müssen, wie z.B. nach einem Neustart oder RESET.

17h *Telegramm zu lang*

Das empfangene Telegramm ist zu lang. Sie können max. 1024Byte an Nutzdaten übertragen

07h *Quittungsverzug*

Das Telegramm wurde innerhalb der Quittungsverzugszeit nicht quittiert.

0Ah *DBL überschritten*

Die Anzahl der Blockwiederholungen, welche Sie im Parameter "Datenblocklänge DBL" angeben können, wurde überschritten.

Statusverwaltung CPU Byte 1

Hier können Sie erkennen, ob ein Auftrag bereits gestartet ist, ob hierbei Fehler aufgetreten sind oder ob der Auftrag gesperrt ist, dass beispielsweise eine virtuelle Verbindung nicht mehr besteht.

Bit 1: Auftrag läuft

Setzen: Durch die Anschaltung, wenn Auftrag an CP erteilt ist.

Löschen: Durch die Anschaltung, wenn ein Auftrag abgearbeitet ist (z.B. Quittung eingetroffen).

Auswerten: Durch die Hantierungsbausteine: Ein neuer Auftrag wird nur erteilt, wenn der "alte" Auftrag abgearbeitet ist.

Durch den Anwender: um zu erfahren, ob das Triggern eines neuen Auftrags sinnvoll ist.

Bit 2: Auftrag fertig ohne Fehler

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag ohne Fehler abgeschlossen wurde.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender zur Prüfung, ob der Auftrag fehlerlos abgeschlossen wurde.

Bit 3: Auftrag fertig mit Fehler

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag mit Fehler abgeschlossen wurde. Die Fehlerursache befindet sich dann in Byte 0 des Anzeigeworts.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender: Zur Prüfung, ob der Auftrag mit Fehler abgeschlossen, wurde. Ist die Kennung "Auftrag fertig mit Fehler" gesetzt, befindet sich in Byte 0 des Anzeigeworts der entsprechende Fehlercode.

ASCII / STX/ETX / 3964(R) / RK512 - Grundlagen

ASCII

Die Datenkommunikation via ASCII ist eine einfache Form des Datenaustauschs und kann mit einer Multicast/Broadcast-Funktion verglichen werden.

Die logische Trennung der Telegramme wird über 2 Zeitfenster gesteuert. Der Sender muss sein Telegramm innerhalb der "Zeichenverzugszeit" (ZVZ), die im Empfänger vorgegeben wird, schicken.

Solange "Zeit nach Auftrag" (ZNA) nicht abgelaufen ist, wird kein neuer Sendeauftrag angenommen.

Mit diesen beiden Zeitangaben kann eine einfach serielle SPS-SPS-Kommunikation aufgebaut werden.

Das Bit FRG (Sende-Freigabe) wird erst dann zurückgesetzt, wenn die Daten gesendet wurden und die ZNA abgelaufen ist.

ASCII-fragmentiert

Unter ASCII wird ein Telegramm erst dann an die CPU weitergereicht, wenn dieses vollständig empfangen wurde. Mit ASCII-fragmentiert haben Sie die Möglichkeit durch Einsatz des Receive-Bausteins FC11 (ASCII_FRAGMENT) große Telegramme in Blöcken direkt nach dem Erhalt an die CPU weiterzureichen. Hierbei beträgt die Blocklänge 12Byte. Bei ASCII-fragmentiert wartet der CP nicht bis das komplette Telegramm empfangen wurde.

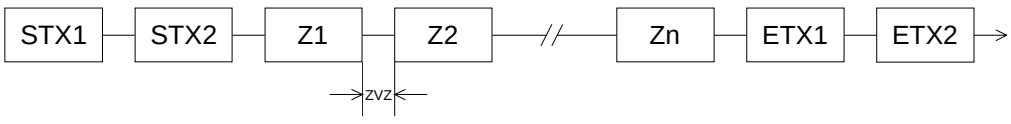
STX/ETX

STX/ETX ist ein einfaches Protokoll mit Header und Trailer. STX/ETX wird zur Übertragung von ASCII-Zeichen (20h…7Fh) eingesetzt. Dies erfolgt ohne Blockprüfung (BCC). Sollen Daten von der Peripherie eingelesen werden, muss als Startzeichen STX (Start of Text) vorhanden sein, anschließend folgen die zu übertragenden Zeichen. Als Schlusszeichen muss ETX (End of Text) vorliegen.

Die Nutzdaten, d.h. alle Zeichen zwischen STX und ETX, werden nach Empfang des Schlusszeichens ETX an die CPU übergeben.

Beim Senden der Daten von der CPU an ein Peripheriegerät werden die Nutzdaten an den CP 240 übergeben und von dort, mit STX als Startzeichen und ETX als Schlusszeichen, an den Kommunikationspartner übertragen.

Telegrammaufbau:



Sie können bis zu 2 Anfangs- und Endezeichen frei definieren. Auch hier kann eine ZNA für den Sender vorgegeben werden.

3964(R)

3964(R) steuert die Datenübertragung bei einer Punkt-zu-Punkt-Kopplung zwischen dem CP 240 und einem Kommunikationspartner. Hier werden bei der Datenübertragung den Nutzdaten Steuerzeichen hinzugefügt. Durch diese Steuerzeichen kann der Kommunikationspartner kontrollieren, ob die Daten vollständig und fehlerfrei bei ihm angekommen sind.

Folgende Steuerzeichen werden ausgewertet:

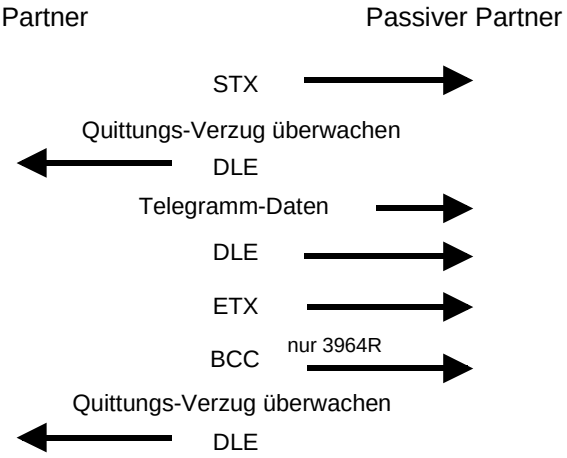
- STX Start of Text
- DLE Data Link Escape
- ETX End of Text
- BCC Block Check Character (nur bei 3964R)
- NAK Negative Acknowledge

Hinweis!

Wird ein DLE als Informationszeichen übertragen, so wird dieses zur Unterscheidung vom Steuerzeichen DLE beim Verbindungsauf- und -abbau auf der Sendeleitung doppelt gesendet (DLE-Verdoppelung). Der Empfänger macht die DLE-Verdoppelung wieder rückgängig.

Unter 3964(R) muss dem Kommunikationspartner eine niedrigere Priorität zugeordnet sein. Wenn beide Kommunikationspartner gleichzeitig einen Sendeauftrag erteilen, dann stellt der Partner mit niedriger Priorität seinen Sendeauftrag zurück.

Ablauf

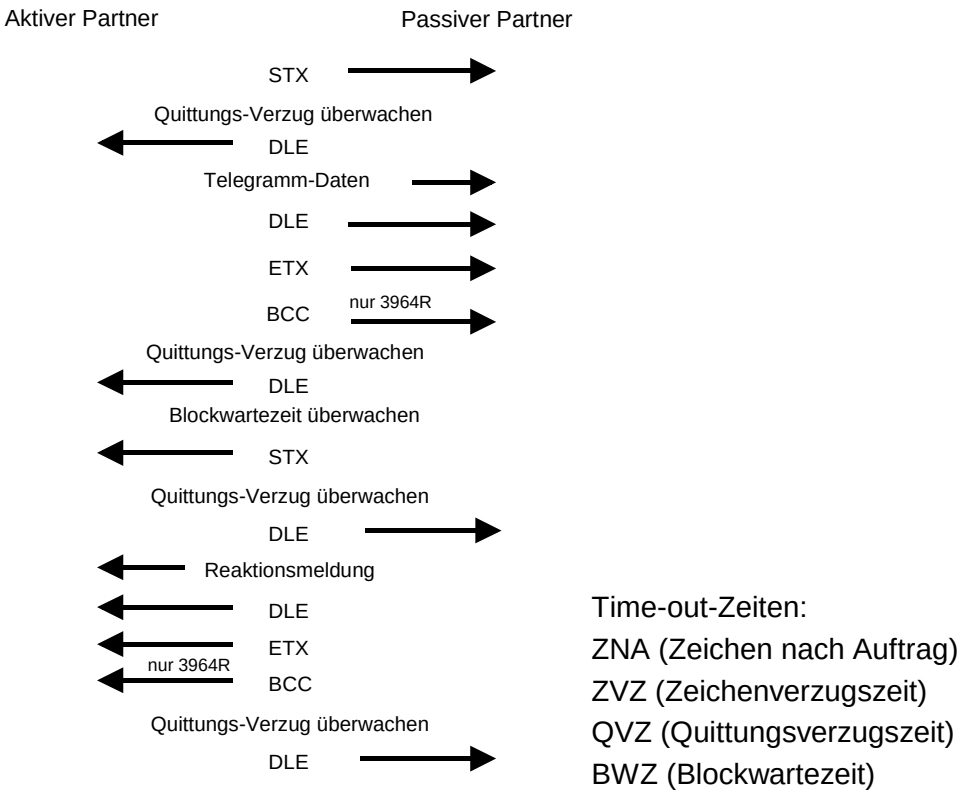


Sie können pro Telegramm maximal 250Byte übertragen.

3964(R) mit RK512

Das RK512 ist ein erweitertes 3964(R). Es wird lediglich vor der Übertragung der Nutzdaten ein Telegrammkopf gesendet. Der Telegrammkopf enthält für den Kommunikationspartner Informationen über Größe, Art und Länge der Nutzdaten.

Ablauf



Time-out-Zeiten

QVZ wird überwacht zwischen STX und DLE sowie zwischen BCC und DLE. ZVZ wird während des gesamten Telegramm-Empfangs überwacht.

Bei Verstreichen der QVZ nach STX wird erneut STX gesendet, nach 5 Versuchen wird ein NAK gesendet und der Verbindungsaufbau abgebrochen. Dasselbe geschieht, wenn nach einem STX ein NAK oder ein beliebiges Zeichen empfangen wird.

Bei Verstreichen der QVZ nach dem Telegramm (nach BCC-Byte) oder bei Empfang eines Zeichens ungleich DLE werden der Verbindungsaufbau und das Telegramm wiederholt. Auch hier werden 5 Versuche unternommen, danach ein NAK gesendet und die Übertragung abgebrochen.

Die Blockwartezeit (BWZ) ist die maximale Zeitdauer zwischen der Bestätigung eines Anforderungstelegrams (DLE) und STX des Reaktionstelegramms. Bei Überschreiten der BWZ wird mehrere Male (über DBL parametrierbar) versucht das Anforderungstelegramm zu senden. Sind diese Versuche erfolglos, wird die Übertragung abgebrochen.

Passivbetrieb

Wenn der Treiber auf den Verbindungsaufbau wartet und ein Zeichen ungleich STX empfängt, sendet er NAK. Bei Empfang eines Zeichens NAK sendet der Treiber keine Antwort.

Wird beim Empfang die ZVZ überschritten, wird ein NAK gesendet und auf erneuten Verbindungsaufbau gewartet.

Wenn der Treiber beim Empfang des STX noch nicht bereit ist, sendet er ein NAK.

Block-Check-Character (BCC-Byte)

Zur weiteren Datensicherung wird bei 3964R am Ende des Telegramms ein **B**lock-**C**heck-**C**harakter angehängt. Das BCC-Byte wird durch eine XOR-Verknüpfung über die Daten des gesamten Telegramms einschließlich DLE/ETX gebildet.

Beim Empfang eines BCC-Bytes, das vom selbst ermittelten abweicht, wird anstatt des DLEs ein NAK gesendet.

Initialisierungs-konflikt

Versuchen beide Partner gleichzeitig innerhalb der QVZ einen Verbindungsaufbau, so sendet der Partner mit der niedrigeren Priorität das DLE und geht auf Empfang.

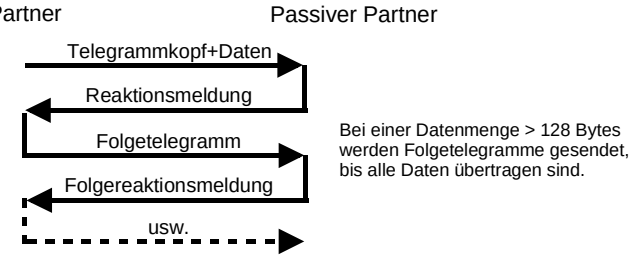
Data Link Escape (DLE-Zeichen)

Das DLE-Zeichen in einem Telegramm wird vom Treiber verdoppelt, d.h. es wird DLE/DLE gesendet. Beim Empfang werden doppelte DLEs als ein DLE im Puffer abgelegt. Als Ende des Telegramms gilt immer die Kombination DLE/ETX/BCC (nur bei 3964R).

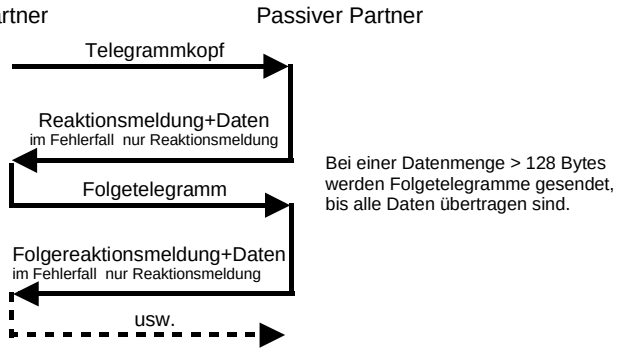
Die Steuercodes : 02h = STX
03h = ETX
10h = DLE
15h = NAK

Logischer Telegrammablauf

SEND (Senden von Daten)



FETCH (Holen von Daten)



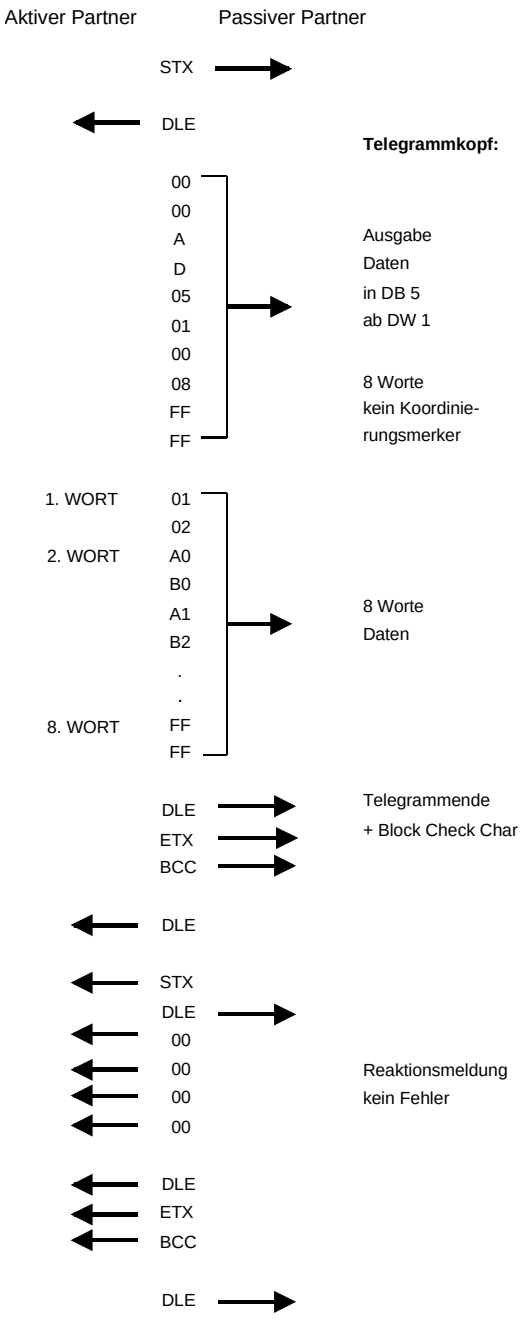
In beiden Fällen wird maximal 5s auf die Reaktionsmeldung gewartet, und danach der Empfang abgebrochen.

Inhalt der Telegramme

Jedes Telegramm besitzt einen Kopf. Je nach der Vorgeschichte des Telegrammverkehrs enthält dieser alle erforderlichen Informationen.

Aufbau Ausgabe-Telegramm

Beispiel Ausgabetelegramm



Normales Telegramm

Byte		
0 1	00 00	Kennung für Telegramm
2 3	A X	Ausgabebefehl Art der Daten
4 5	xx xx	Parameter 1 Ziel
6 7	yy yy	Parameter 2 Anzahl
8 9	zz zz	Parameter 3 Koordinierungsmerker
10 - N	aa bb xy	Daten

mit N = 10 ... 127

Reaktionstelegramm

Byte		
0 1 2	00 00 00	Kennung für Reaktions-meldung
3	xx	Fehlercode

Bei Datenmengen >128Byte werden Folgetelegramme gesendet.

Aufbau Folgetelegramme

Folgetelegramm

Byte		
0 1	FF 00	Kennung für Folgetelegramm
2 3	A X	Ausgabebefehl Art der Daten
4 - N	aa bb xy	Daten

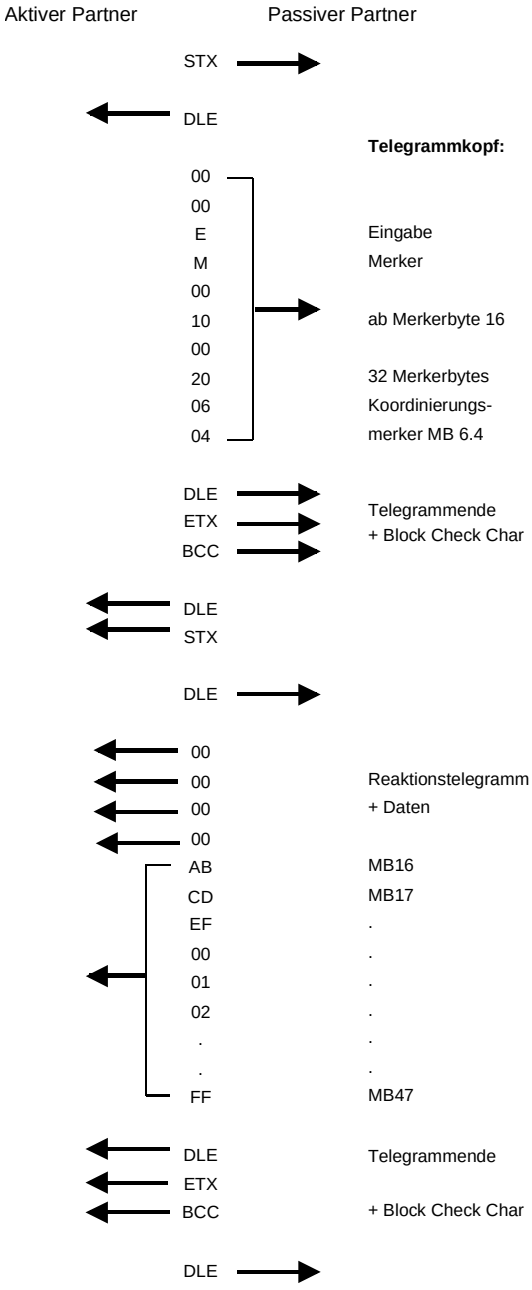
mit N = 4 ... 127

Folge-Reaktionstelegramm

Byte		
0 1 2	FF 00 00	Kennung für Folge-Reaktions-telegramm
3	xx	Fehlercode

Aufbau Eingabe-Telegramm

Beispiel Eingabetelegramm



Normales Telegramm

Byte		
0	00	Kennung für
1	00	Telegramm
2	E	Eingabebefehl
3	X	Art der Daten
4	xx	Parameter 1
5	xx	Ziel
6	yy	Parameter 2
7	yy	Anzahl
8	zz	Parameter 3
9	zz	Koordinierungsmerker

Reaktionstelegramm

Byte		
0	00	Kennung für
1	00	Reaktions-
2	00	meldung
3	xx	Fehlercode
4	aa	
-	bb	Daten
N	xy	

mit N = 4 ... 127

Bei Datenmengen > 128Byte werden Folgetelegramme gesendet.

Aufbau Folgetelegramme

Folgetelegramm

Byte		
0	FF	Kennung für
1	00	Folgetelegramm
2	E	Eingabebefehl
3	X	Art der Daten

Folge-Reaktionstelegramm

Byte		
0	FF	Kennung für
1	00	Folge-Reaktions-
2	00	telegramm
3	xx	Fehlercode
4	aa	
-	bb	Daten
N	xy	

mit N = 4 ... 127

Koordinierungs-merker

Der Koordinierungsmerker wird im Aktiv-Betrieb im Partner-AG bei Empfang eines Telegramms gesetzt. Dies geschieht sowohl bei Eingabe- als auch bei Ausgabe-Befehlen. Ist der Koordinierungsmerker gesetzt und wird ein Telegramm mit diesem Merker empfangen, so werden die Daten nicht übernommen (bzw. übergeben), sondern es wird eine Fehler-Reaktionsmeldung gesendet (Fehlercode 32h). In diesem Fall muss der Koordinierungsmerker vom Anwender im Partner-AG zurückgesetzt werden.

ASCII / STX/ETX / 3964(R) / RK512 - Kommunikationsprinzip

Kommunikation über Hantierungs-bausteine

Die serielle Kommunikation erfolgt unter Einsatz von Hantierungs-bausteinen. Diese Hantierungsbausteine finden Sie im Service-Bereich unter www.vipa.com.

Je nach Protokoll kommen folgende Hantierungsbausteine zum Einsatz:

ASCII	STX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

Hinweis!

Eine Kommunikation mit SEND- und RECEIVE-Bausteinen ist nur möglich, wenn zuvor im Anlauf-OB der Paramater ANL des SYNCHRON-Bausteins gesetzt wurde.

Daten senden und empfangen

Daten, die von der CPU über den Rückwandbus in den entsprechenden Datenkanal geschrieben werden, werden vom Kommunikationsprozessor in den entsprechenden Sendepuffer (1024Byte) geschrieben und von dort über die Schnittstelle ausgegeben.

Empfängt der Kommunikationsprozessor Daten über die Schnittstelle, werden die Daten in einem Ringpuffer abgelegt (1024Byte). Die empfangenen Daten können über den Datenkanal von der CPU gelesen werden.

Kommunikation über Rück-wandbus

Der Austausch von empfangenen Telegrammen über den Rückwandbus erfolgt asynchron. Ist ein komplettes Telegramm über die serielle Schnittstelle eingetroffen (Ablauf der ZVZ), so wird dies in einem 1024Byte großen Ringpuffer abgelegt. Aus der Länge des noch freien Ringpuffers ergibt sich die max. Länge eines Telegramms. Je nach Parametrierung können bis zu 250 Telegramme gepuffert werden, wobei deren Gesamtlänge 1024 nicht überschreiten darf.

Ist der Puffer voll, werden neu ankommende Telegramme verworfen. Ein komplettes Telegramm wird in je 12Byte große Blöcke unterteilt und an den Rückwandbus übergeben. Das Zusammensetzen der Datenblöcke hat in der CPU zu erfolgen.

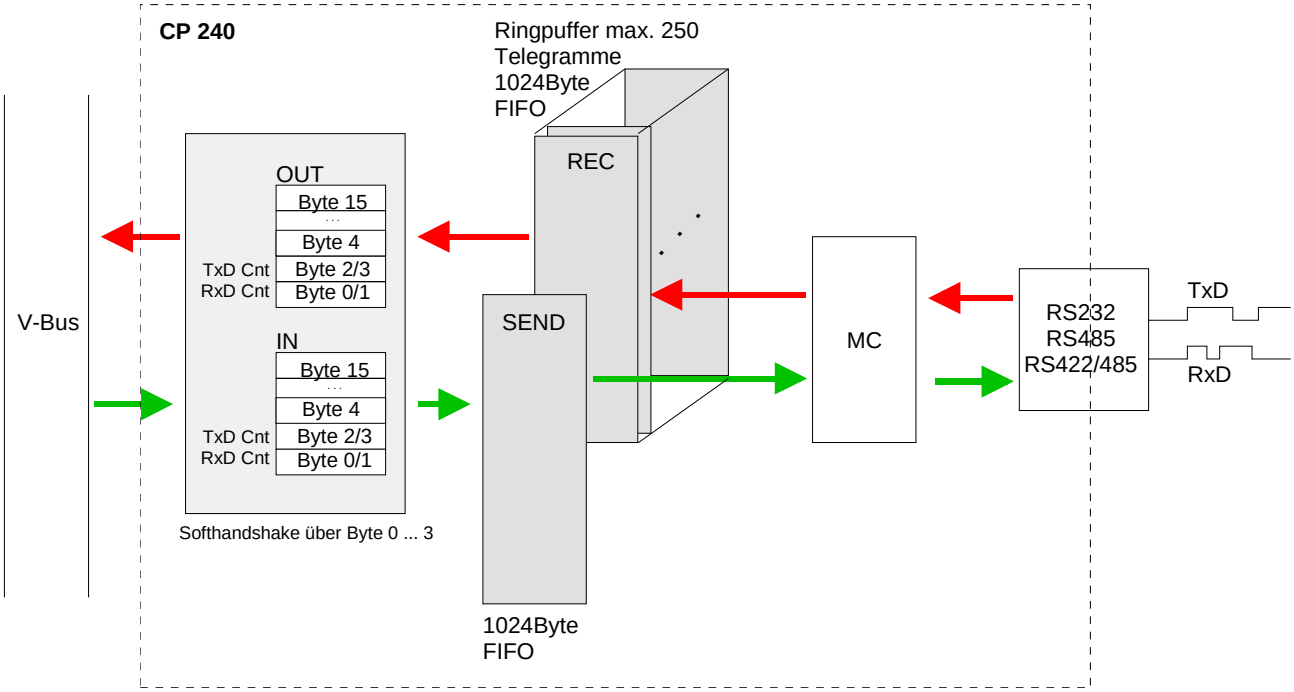
Kommunikation unter ASCII-fragmentiert

Unter ASCII-fragmentiert werden ankommende Daten eines Telegramms sofort in Blöcken an die CPU weitergereicht. Hierbei beträgt die Blocklänge mindestens 12Byte. Bei ASCII-fragmentiert wartet der CP nicht bis das komplette Telegramm empfangen wurde.

Aufgaben der CPU

Ein zu sendendes Telegramm ist in der CPU in 12Byte große Blöcke zu unterteilen und über den Rückwandbus an den CP 240 zu übergeben. Im CP 240 werden diese Blöcke im Sendepuffer zusammengesetzt und bei Vollständigkeit des Telegramms über die serielle Schnittstelle gesendet. Da der Datenaustausch über den Rückwandbus asynchron abläuft, wird ein "Software Handshake" zwischen dem CP 240 und der CPU eingesetzt. Die Register für den Datentransfer vom CP 240 sind 16Byte breit. Für den Handshake sind die Bytes 0 bis 3 (Wort 0 und 2) reserviert.

Folgende Abbildung soll dies veranschaulichen:



Software-handshake

Für den Einsatz des CP 240 in Verbindung mit einer System 200V CPU sind bei VIPA Hantierungsbausteine erhältlich, die den Softwarehandshake komfortabel übernehmen.

Bei Einsatz des CP 240 ohne Hantierungsbausteine soll hier die Funktionsweise anhand eines Beispiels für das Senden und Empfangen von Daten erläutert werden.

Beispiel Daten senden

Es soll z.B. ein Telegramm mit der Länge von 30Byte gesendet werden. So werden von der CPU die ersten 12Byte Nutzdaten des Telegramms in die Bytes 4 bis 15 und in Byte 2/3 die Länge des Telegramms (also "30") geschrieben. Der CP 240 empfängt die Daten über den Rückwandbus und kopiert die 12Byte Nutzdaten in den Sendepuffer. Zur Quittierung des Empfangs schreibt der CP 240 in Byte 2/3 den Wert "30" (Länge des Telegramms) zurück.

Beim Empfang der "30", kann die CPU weitere 12Byte Nutzdaten in Byte 4 bis 15 und die Restlänge des Telegramms ("18" Byte) in Byte 2/3 an den CP 240 senden. Dieser speichert wieder die Nutzdaten im Sendepuffer und gibt die Restlänge des Telegramms ("18") auf Byte 2/3 an die CPU zurück.

Beim Empfang der "18", kann die CPU die restlichen 6Byte Nutzdaten in den Byte 4 bis 9 und die Restlänge des Telegramms (also "6") in Byte 2/3 an den CP 240 senden. Dieser speichert die Nutzdaten im Sendepuffer ab und schreibt den Wert "6" auf Byte 2/3 an die CPU zurück.

Beim Empfang der "6" auf Byte 2/3 sendet die CPU eine "0" auf Byte 2/3. Der CP 240 stößt daraufhin das Senden des Telegramms über die serielle Schnittstelle an und schreibt, wenn alle Daten übertragen sind, eine "0" auf Byte 2/3 zurück.

Beim Empfang der "0" kann die CPU ein neues Telegramm an den CP 240 senden.

Beispiel Daten empfangen

Die Schnittstelle des CP 240 hat z.B. ein Telegramm mit 18Byte Länge über die serielle Schnittstelle empfangen. Der CP 240 schreibt die ersten 12Byte Nutzdaten in die Bytes 4 bis 15 des Empfangspuffers und in Byte 0/1 die Länge des Telegramms (also "18"). Die Daten werden über den Rückwandbus an die CPU übertragen. Die CPU speichert die 12Byte Nutzdaten und sendet den Wert "18" auf Byte 0/1 an den CP 240 zurück.

Beim Empfang der "18", schreibt der CP 240 die restlichen 6Byte Nutzdaten in die Byte 4 bis 9 des Empfangspuffers und in Byte 0/1 die Länge ("6") der übergebenen Nutzdaten. Die CPU speichert die Nutzdaten und gibt an den CP 240 in Byte 0/1 den Wert "6" zurück.

Beim Empfang der "6" sendet der CP 240 den Wert "0" auf Byte 0/1, für Telegramm komplett, an die CPU zurück. Die CPU sendet eine "0" auf Byte 0/1 an den CP 240 zurück.

Mit dem Empfang der "0" kann der CP 240 ein neues Telegramm an die CPU senden.

ASCII / STX/ETX / 3964(R) / RK512 - Parametrierung

Allgemein

Sie können dem CP 240 zur Parametrierung 16Byte Parameterdaten übergeben. Der Aufbau der Parameterdaten richtet sich nach dem gewählten Protokoll.

Bei der Hardware-Konfiguration ist immer der dem Protokoll entsprechende CP 240 zu verwenden.

Nachfolgend finden Sie eine Auflistung der Parameterbytes mit ihren Default-Werten.

Aufbau Parameterbytes bei ASCII

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	01h: ASCII 11h: ASCII-fragmentiert	01h: (ASCII)
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine
3	reserviert	0	0
4	ZNA (*20ms)	0 ... 255	0
5	ZVZ (*20ms)	0 ... 255	10
6	Anz.Receivebuffer	1 ... 250	1
7...15	reserviert		

Aufbau Parameterbytes bei STX/ETX

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	02h: STX/ETX	02h: (STX/ETX)
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine
3	reserviert	0	0
4	ZNA (*20ms)	0 ... 255	0
5	TMO (*20ms)	1 ... 255	10
6	Anzahl Startkennungen	0 ... 2	01
7	Startkennung 1	0 ... 255	02
8	Startkennung 2	0 ... 255	0
9	Anzahl Endekennungen	0 ... 2	01
10	Endekennung 1	0 ... 255	03
11	Endekennung 2	0 ... 255	0
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Aufbau Parameterbytes bei 3964(R) / 3964(R) mit RK512

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	03h: 3964 04h: 3964R 05h: 3964 + RK512 06h: 3964R + RK512	03h: 3964
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-
3	reserviert	0	0
4	ZNA (*20ms)	0 ... 255	0
5	ZVZ (*20ms)	0 ... 255	10
6	QVZ (*20ms)	0 ... 255	25
7	BWZ (*20ms)	0 ... 255	100
8	STX Wiederholungen	0 ... 255	5
9	DBL	0 ... 255	6
10	Priorität	0: low 1: high	0: low
11	reserviert		
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Parameter-beschreibung

Baudrate

Geschwindigkeit der Datenübertragung in Bit/s (Baud).
Sie haben folgende Einstellmöglichkeiten:

00h:	Default (9600Baud)
01h:	150Baud
02h:	300Baud
03h:	600Baud
04h:	1200Baud
05h:	1800Baud
06h:	2400Baud
07h:	4800Baud
08h:	7200Baud
09h:	9600Baud
0Ah:	14400Baud
0Bh:	19200Baud
0Ch:	38400Baud
0Dh:	57600Baud
0Fh:	76800Baud
0Eh:	115200Baud

Default: 0 (9600Baud)

Protokoll

Das Protokoll, das verwendet werden soll. Diese Einstellung beeinflusst den weiteren Aufbau der Parameterdaten.
Sie haben folgende Einstellmöglichkeiten:

01h:	ASCII
02h:	STX/ETX
03h:	3964
04h:	3964R
05h:	3964 und RK512
06h:	3964R und RK512
11h:	ASCII-fragmentiert

Übertragungs-parameter-Byte

Für jeden Zeichenrahmen stehen je 3 Datenformate zur Verfügung. Die Datenformate unterscheiden sich durch Anzahl der Datenbits, mit oder ohne Paritätsbit und Anzahl der Stopbits.

Das Übertragungsparameter-Byte hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine

Datenbits

Anzahl der Datenbits, auf die ein Zeichen abgebildet wird.

Parity

Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet.

Stopbits

Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens.

Flusskontrolle
(bei ASCII und STX/ETX)

Mechanismus, der den Datentransfer synchronisiert, wenn der Sender schneller Daten schickt als der Empfänger verarbeiten kann. Die Flusskontrolle kann hardware- oder softwaremäßig (XON/XOFF) erfolgen. Bei der Hardware-Flusskontrolle werden die Leitungen RTS und CTS verwendet, die dann entsprechend zu verdrahten sind.

Die Software-Flusskontrolle verwendet zur Steuerung die Steuerzeichen XON=11h und XOFF=13h. Bitte beachten Sie, dass dann Ihre Daten diese zwei Steuerzeichen nicht beinhalten dürfen.

Default: 13h (Datenbits: 8, Parität: keine, Stopbit: 1, Flusskontrolle: keine)

Zeit nach Auftrag (ZNA)

Wartezeit, die eingehalten wird, bis der nächste Sendeauftrag ausgeführt wird. Die ZNA wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 0*

Zeichenverzugszeit (ZVZ)
(bei ASCII, 3964(R) und RK512)

Die Zeichenverzugszeit definiert den maximal zulässigen zeitlichen Abstand zwischen zwei empfangenen Zeichen innerhalb eines Telegramms. Die ZVZ wird in 20ms-Einheiten angegeben.

Bei ZVZ=0 berechnet sich der CP anhand der Baudrate die ZVZ selbst (ca. doppelte Zeichenzeit).

Bereich: 0 ... 255 *Default: 10*

Anzahl Receivebuffer
(nur bei ASCII)

Legt die Anzahl der Empfangspuffer fest. Solange nur 1 Empfangspuffer verwendet wird und dieser belegt ist, können keine weiteren Daten empfangen werden. Durch Aneinanderreihung von bis zu 250 Empfangspuffern können die empfangenen Daten in einen noch freien Empfangspuffer umgeleitet werden.

Bereich: 1 ... 250 *Default: 1*

Time-out (TMO)
(nur bei STX/ETX)

Mit TMO definieren Sie den maximal zulässigen zeitlichen Abstand zwischen zwei Telegrammen. TMO wird in 20ms-Einheiten angegeben.

Bereich: 1 ... 255 *Default: 10*

Anzahl Startkennungen
(nur bei STX/ETX)

Hier können Sie 1 oder 2 Startkennungen einstellen. Ist "1" als Anzahl der Startkennungen eingestellt, wird der Inhalt des 2. Startkennzeichens (Byte 8) ignoriert.

Bereich: 0 ... 2 *Default: 1*

Startkennung 1 und 2 (STX)
(nur bei STX/ETX)

ASCII-Wert des Startzeichens, das einem Telegramm vorausgeschickt wird und den Start einer Übertragung kennzeichnet. Sie können 1 oder 2 Startzeichen verwenden. Bei Einsatz von 2 Startzeichen müssen Sie unter "Anzahl Startkennungen" eine 2 eintragen.

Startkennung 1, 2: *Bereich: 0 ... 255* *Default: 2 (Kennung 1)*
0 (Kennung 2)

Anzahl Endekennungen
(nur bei STX/ETX)

Hier können Sie 1 oder 2 Endekennungen einstellen. Ist "1" als Anzahl der Endekennungen eingestellt, wird der Inhalt des 2. Endekennzeichens (Byte 11) ignoriert.

Bereich: 0 ... 2 *Default: 1*

Endekennung 1 und 2 (ETX)
(nur bei STX/ETX)

ASCII-Wert des Endezeichens, das nach einem Telegramm folgt und das Ende einer Übertragung kennzeichnet. Sie können 1 oder 2 Endezeichen verwenden. Bei Einsatz von 2 Endezeichen müssen Sie unter "Anzahl Endekennungen" eine 2 eintragen.

Endekennung 1, 2: *Bereich: 0 ... 255* *Default: 3 (Kennung 1)*
0 (Kennung 2)

Quittungs-verzugszeit (QVZ)
(bei 3964(R), RK512)

Die Quittungsverzugszeit definiert den maximal zulässigen zeitlichen Abstand bis zur Quittung des Partners bei Verbindungsauf- und -abbau. Die QVZ wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 25*

Blockwartezeit (BWZ)
(bei 3964(R), RK512)

Die Blockwartezeit (BWZ) ist die maximale Zeitdauer zwischen der Bestätigung eines Anforderungstelegrams (DLE) und STX des Reaktions-telegramms. Die BWZ wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 100*

STX-Wiederholungen
(bei 3964(R), RK512)

Maximale Anzahl der Versuche des CP 240 eine Verbindung aufzubauen.

Bereich: 0 ... 255 *Default: 3*

Wiederholung Datenblöcke bei BWZ-Über-schreitung (DBL)
(bei 3964(R), RK512)

Bei Überschreiten der Blockwartezeit (BWZ) können Sie über den Parameter DBL die maximale Anzahl Wiederholungen für das Anforderungstelegramm vorgeben. Sind diese Versuche erfolglos, wird die Übertragung abgebrochen.

Bereich: 0 ... 255 *Default: 6*

Priorität
(bei 3964(R), RK512)

Ein Kommunikationspartner hat hohe Priorität, wenn sein Sendeversuch Vorrang gegenüber dem Sendewunsch des Partners hat. Bei niedriger Priorität muss dieser hinter dem Sendewunsch des Partners zurückstehen.

Bei den Protokollen 3964(R) und RK512 müssen die Prioritäten beider Partner unterschiedlich sein.

Sie haben folgende Einstellmöglichkeiten:

0: low

1: high *Default: 0 (low)*

Modbus - Grundlagen

Übersicht

Das Protokoll Modbus ist ein Kommunikationsprotokoll, das eine hierarchische Struktur mit einem Master und mehreren Slaves festlegt.

Master-Slave-Kommunikation

Es treten keine Buskonflikte auf, da der Master immer nur mit einem Slave kommunizieren kann. Nach einer Anforderung vom Master wartet dieser solange auf die Antwort des Slaves, bis eine einstellbare Wartezeit abgelaufen ist. Während des Wartens ist eine Kommunikation mit einem anderen Slave nicht möglich.

Telegramm-Aufbau

Die Anforderungs-Telegramme, die ein Master sendet, und die Antwort-Telegramme eines Slaves haben den gleichen Aufbau:

Start-zeichen	Slave-Adresse	Funktions-Code	Daten	Fluss-kontrolle	Ende-zeichen

Broadcast mit Slave-Adresse = 0

Eine Anforderung kann an einen bestimmten Slave gerichtet sein oder als Broadcast-Nachricht an alle Slaves gehen. Zur Kennzeichnung einer Broadcast-Nachricht wird die Slave-Adresse 0 eingetragen.

Nur Schreibaufträge dürfen als Broadcast gesendet werden.

ASCII-, RTU-Modus

Bei Modbus gibt es zwei unterschiedliche Übertragungsmodi:

- ASCII-Modus: Jedes Byte wird im 2 Zeichen ASCII-Code übertragen. Die Daten werden durch Anfang- und Ende-Zeichen gekennzeichnet. Dies macht die Übertragung transparent, aber auch langsam.
- RTU-Modus: Jedes Byte wird als ein Zeichen übertragen. Hierdurch erreichen Sie einen höheren Datendurchsatz als im ASCII-Modus. Anstelle von Anfang- und Ende-Zeichen wird eine Zeitüberwachung eingesetzt.

Die Modus-Wahl erfolgt bei der Parametrierung.

Modbus auf dem CP 240 von VIPA

Der CP 240 Modbus unterstützt verschiedene Betriebsarten, die nachfolgend beschrieben sind:

Modbus Master

Im *Modbus Master* Betrieb steuern Sie die Kommunikation über Ihr SPS-Anwenderprogramm. Hierzu sind die SEND- und RECEIVE-Hantierungsbausteine erforderlich. Sie haben hier die Möglichkeit unter Verwendung einer Blockung bis zu 250Byte Nutzdaten zu übertragen.

Modbus Slave Short

Im *Modbus Slave Short* Betrieb belegt der CP 240 je 16Byte für Ein- und Ausgabe-Daten an beliebiger Stelle in der CPU. Über die Adress-Parameter können Sie bei der Hardware-Konfiguration diesen Bereich definieren. Ein SPS-Programm für die Datenbereitstellung ist auf Slave-Seite nicht erforderlich. Diese Betriebsart eignet sich besonders zur schnellen Datenübertragung kleiner Datenmengen über Modbus

Modbus Slave Long

Für Daten, deren Länge 16Byte überschreiten, sollten Sie die Modbus Slave Long Betriebsart verwenden. Hier wird bei Datenempfang vom Master mit RECEIVE der Bereich an die CPU übergeben, innerhalb dessen eine Änderung stattgefunden hat. Der Datentransfer erfolgt nach folgendem Prinzip:

Der max. 1024Byte große Empfangsbereich wird in 128 8Byte-Blöcke aufgeteilt. Bei Datenänderung durch den Master werden nur die Blöcke an die CPU übergeben, in denen geändert wurde. In einem Baustein-Zyklus des RECEIVE-Bausteins können maximal 16 zusammenhängende 8Byte-Blöcke am Rückwandbus übergeben werden. Liegen die 8Byte-Blöcke nicht zusammen, ist für jeden geänderten 8Byte-Block ein Baustein-Zyklus erforderlich. Der Empfangs-DB des RECEIVE-Bausteins ist immer als ein Vielfaches von 8 anzugeben.

Mit einem SEND-Aufruf wird ein gewünschter Datenbereich in den CP übertragen, der vom Master gelesen werden kann. Schreibende Master-Zugriffe dürfen nicht außerhalb des Empfangs-Bereichs liegen!

Bitte beachten Sie, dass Modbus Slave Long ab der Baustein-Bibliothek FX000002_V120 oder höher unterstützt wird.

Hinweis!

Erst nachdem alle Daten im CP 240 vorliegen, sendet der CP 240 ein Antworttelegramm an den Master.

Inbetriebnahme

Nach Einschalten der Spannungsversorgung leuchten am Modbus-Modul die LEDs ER, TxD und RxD. Das Modul signalisiert hiermit, dass es noch keine gültigen Parameter von der CPU erhalten hat. Sobald Sie die CPU in RUN schalten, werden die Modbus-Parameter an das Modul übertragen. Bei gültigen Parametern erlöschen die LEDs ER, TxD und RxD. Das Modbus-Modul ist nun bereit für die Kommunikation.

Bei Einsatz im Master-Modus können Sie nun entsprechende Schreib-/Lesebefehle in Ihrem Anwenderprogramm ausführen lassen.

Sollte die ER-LED nicht erlöschen, liegt ein interner Fehler vor. Bei einem vorübergehenden Fehler können Sie diesen durch einen STOP-RUN-Übergang der CPU rücksetzen.

Modbus - Parametrierung

Aufbau Parameter bei Modbus

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	0h: 9600Baud 6h: 2400Baud 7h: 4800Baud 9h: 9600Baud Ah: 14400Baud Bh: 19200Baud Ch: 38400Baud	0h: 9600Baud
1	Protokoll	0Ah: Modbus Master ASCII short 0Bh: Modbus Master RTU short 0Ch: Modbus Slave ASCII short 0Dh: Modbus Slave RTU short 1Ch: Modbus Slave ASCII long 1Dh: Modbus Slave RTU long	Bh: Modbus Master RTU
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-
3	reserviert	0	0
4	Adresse	1...255	1
5	Debug	0: Debug aus 1: Debug ein	0
6...7	Wartezeit	0: automat. Berechnung 1 ... 60000: Zeit in ms	0
8	reserviert		
9	reserviert		
10	reserviert		
11	reserviert		
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Hinweis zu den Defaultparametern!

Sofern keine Parametrierung vorhanden ist und der CP 240 über Auto-Adressierung eingebunden werden soll, besitzt der CP folgende Default-Parameter:

Baudrate: 9600Baud, Protokoll: ASCII, Datenbits: 8, **Parity: even**, Stopbits: 1, Flusskontrolle: keine, ZNA: 0, ZVZ: 200ms, Receivebuffer: 1

Parameter-beschreibung

Baudrate

Geschwindigkeit der Datenübertragung in Bit/s (Baud).
Sie haben folgende Einstellmöglichkeiten:

00h: Default (9600Baud)
06h: 2400Baud
07h: 4800Baud
09h: 9600Baud
0Ah: 14400Baud
0Bh: 19200Baud
0Ch: 38400Baud

Default: 0 (9600Baud)

Protokoll

Das Protokoll, das verwendet werden soll. Diese Einstellung beeinflusst den weiteren Aufbau der Parameterdaten.

0Ah: Modbus Master mit ASCII
0Bh: Modbus Master mit RTU
0Ch: Modbus Slave Short mit ASCII
0Dh: Modbus Slave Short mit RTU
1Ch: Modbus Slave Long mit ASCII
1Dh: Modbus Slave Long mit RTU

Übertragungs-parameter-Byte

Für jeden Zeichenrahmen stehen je 3 Datenformate zur Verfügung. Die Datenformate unterscheiden sich durch Anzahl der Datenbits, mit oder ohne Paritätsbit und Anzahl der Stopbits.
Das Übertragungsparameter-Byte hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-

Datenbits — Anzahl der Datenbits, auf die ein Zeichen abgebildet wird.

Parity — Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet.

Stopbits — Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens.

Default: 13h (Datenbits: 8, Parität: keine, Stopbit: 1)

Adresse — Stellen Sie hier im Slave-Modus die Modbus-Slave-Adresse ein.
Bereich: 1 ... 255 *Default: 1*

Debug — Dieser Modus ist für interne Tests. Diese Funktion sollte immer deaktiviert sein.
Bereich: 0, 1 *Default: 0*

Wartezeit — Hier ist im Master-Modus eine Wartezeit in ms vorzugeben. Mit "0" wird die Wartezeit protokollabhängig nach folgender Formel automatisch ermittelt:

Modbus ASCII: $50ms + \frac{2926000ms}{Baudrate} \cdot s$ mit Baudrate in Bit/s

Modbus RTU: $50ms + \frac{5190000ms}{Baudrate} \cdot s$ mit Baudrate in Bit/s

Im Slave-Modus wird dieser Parameter ignoriert.

Modbus - Einsatz

Übersicht

Sie können den CP 240 Modbus sowohl im Master- als auch im Slave-Modus betreiben. In beiden Modi belegt das Modul für Ein- und Ausgangs-Daten je 16Byte an beliebiger Stelle in der CPU.

Für den Einsatz unter Modbus ist immer eine Hardware-Konfiguration durchzuführen.

Voraussetzung für den Betrieb

Folgende Komponenten sind zum Einsatz der System 200V Modbus-Module erforderlich:

- Je 1 System 200V bestehend aus CPU 21x und CP 240
- Siemens SIMATIC Manager
- Programmierkabel für MPI-Kopplung (z.B. Green Cable von VIPA)
- GSD-Datei **VIPA_21x.gsd** (V1.67 oder höher)
- VIPA Hantierungsbausteine Vipa_Bibliothek_Vxxx.zip
- Serielle Verbindung zwischen beiden CP 240

Parametrierung

Für den CP 240 ist immer eine Hardware-Konfiguration durchzuführen. Hierfür ist die Einbindung der VIPA_21x.gsd im Hardware-Katalog erforderlich. Die Parametrierung erfolgt nach folgender Vorgehensweise:

- Starten Sie den Siemens SIMATIC Manager
- Installieren Sie die GSD-Datei **VIPA_21x.gsd** im Hardware-Katalog.
- Legen Sie im Hardware-Konfigurator mit der CPU 315-2DP (6ES7 315-2AF03 V1.2) ein virtuelles PROFIBUS-System an.
- Binden Sie an dieses System das Slave-System "VIPA_CPU21x" an und geben Sie diesem die PROFIBUS-Adresse 1.
- Projektieren Sie beginnend mit der CPU 21x Ihr System 200V. Verwenden Sie einen mit "Modbus" bezeichneten CP.
- Parametrieren Sie den CP 240 nach Ihren Vorgaben. Der CP 240 belegt in der CPU je 16Byte für Ein- und Ausgabe.
- Übertragen Sie Ihr Projekt in die SPS.

SPS-Programm

Mit Ausnahme bei "Modbus Slave Short" ist immer für die Kommunikation zusätzlich ein SPS-Programm erforderlich. Hierbei erfolgt die Kommunikation über Hantierungsbausteine, die Sie in Form der VIPA-Bibliothek **Vipa_Bibliothek_*Vxxx.zip*** im Siemens SIMATIC Manager einbinden können. Die Library finden Sie unter www.vipa.com.

Hinweis!

Näheres zur Installation der GSD-Datei und der Library finden Sie im Teil "Projektierung".

Kommunikationsmöglichkeiten

Nachfolgend sollen die Kommunikations-Möglichkeiten zwischen Modbus Master und Modbus Slave an folgenden Kombinationsmöglichkeiten gezeigt werden:

- CP 240 Modbus Master ↔ CP 240 Modbus Slave Short
- CP 240 Modbus Master ↔ CP 240 Modbus Long

Master ↔ Slave Short

Modbus Master

Die Kommunikation im Master-Modus erfolgt über Datenbausteine unter Einsatz der CP 240 SEND-RECEIVE-Hantierungsbausteine. Hier können unter Einsatz einer Blockung bis zu 250Byte Nutzdaten übertragen werden.

Modbus Slave Short

Im *Modbus Slave Short* Modus ist die Anzahl der Nutzdaten für Ein- und Ausgabe auf 16Byte begrenzt. Hierbei ist für den Einsatz auf Slave-Seite lediglich eine Hardware-Konfiguration durchzuführen.

Vorgehensweise

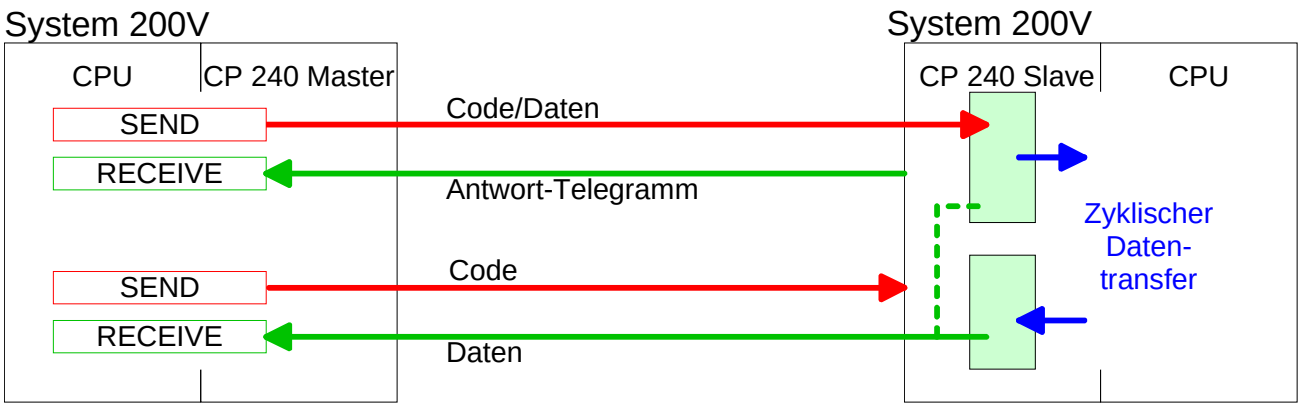
- Bauen Sie für Master- und Slave-Seite je ein System 200V bestehend aus jeweils einer CPU 21x und einem CP 240 auf und verbinden Sie beide Systeme über die serielle Schnittstelle.
- Projektieren Sie die Master-Seite.

  Die Parametrierung des CP 240 als Modbus-Master erfolgt über die Hardware-Konfiguration. Zusätzlich ist für die Kommunikation ein SPS-Anwenderprogramm erforderlich, das nach folgender Struktur aufgebaut sein sollte:

  OB 1: Aufruf des FC0 (SEND) mit Fehlerauswertung. Hierbei ist das Telegramm gemäß den Modbus-Vorgaben im Sendebaustein abzulegen.

  Aufruf des FC1 (RECEIVE) mit Fehlerauswertung. Gemäß den Modbus-Vorgaben werden die Daten im Empfangsbaustein abgelegt.
- Projektieren Sie die Slave-Seite

  Die Parametrierung des CP 240 erfolgt über die Hardware-Konfiguration. Geben Sie hier für Ein- und Ausgabe-Bereich die Startadresse an, ab welcher die fixe Anzahl von 16Byte für Ein- und Ausgabe an beliebiger Stelle in der CPU abliegen.



Master ↔ Slave Long

Modbus Master

Die Kommunikation im Master-Modus erfolgt über Datenbausteine unter Einsatz der CP 240 SEND-RECEIVE-Hantierungsbausteine. Hier können unter Einsatz einer Blockung bis zu 250Byte Nutzdaten übertragen werden.

Modbus Slave Long

Im *Modbus Slave Long* Modus wird nur ein geänderter Datenbereich beginnend bei 0 mit RECEIVE an die CPU übertragen. Fordert der Master Daten an, so ist dafür Sorge zu tragen, dass sich die relevanten Daten im CP befinden. Mit einem SEND-Aufruf wird ein gewünschter Datenbereich von 0 beginnend in den CP übertragen.

Vorgehensweise

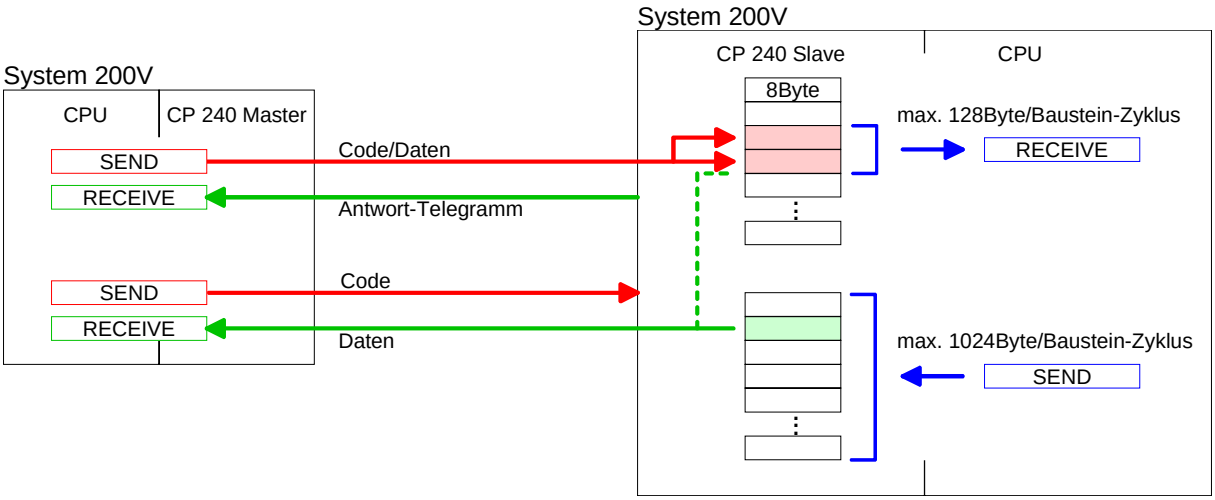
- Bauen Sie für Master- und Slave-Seite je ein System 200V bestehend aus jeweils einer CPU 21x und einem CP 240 auf und verbinden Sie beide Systeme über die serielle Schnittstelle.
- Projektieren Sie die Master-Seite.

  Die Projektierung auf der Master-Seite erfolgt auf die gleiche Weise, wie im Beispiel weiter oben beschrieben.
- Projektieren Sie die Slave-Seite

  Die Parametrierung des CP 240 als Modbus-Slave erfolgt über die Hardware-Konfiguration. Zusätzlich ist für die Kommunikation ein SPS-Anwenderprogramm erforderlich, das nach folgender Struktur aufgebaut sein sollte:

  OB 1: Aufruf des FC0 (SEND) mit Fehlerauswertung. Hierbei wird ein Bereich von 0 beginnend im CP 240 abgelegt, auf den vom Master über Modbus zugegriffen werden kann.

  Mit dem FC1 (RECEIVE) mit Fehlerauswertung können Sie einen Datenbereich in die CPU übertragen. Der max. 1024Byte große Empfangsbereich wird in 128 8Byte-Blöcke aufgeteilt. Bei Datenänderung durch den Master werden nur die Blöcke an die CPU übergeben, in denen geändert wurde. In einem Baustein-Zyklus des RECEIVE-Bausteins können maximal 16 zusammenhängende 8Byte-Blöcke am Rückwandbus übergeben werden. Liegen die 8Byte-Blöcke nicht zusammen, ist für jeden geänderten 8Byte-Block ein Baustein-Zyklus erforderlich. Der Empfangs-DB ist bei Aufruf des RECEIVE-Bausteins immer als ein Vielfaches von 8 anzugeben. Schreibende Master-Zugriffe dürfen nicht außerhalb des Empfangs-Bereichs liegen!



Zugriff auf mehrere Slaves

Bei Einsatz mehrerer Slaves können unter RS485 keine Buskonflikte auftreten, da der Master immer nur mit einem Slave kommunizieren kann. Der Master schickt an den über die Adresse spezifizierten Slave ein Kommandotelegramm und wartet eine gewisse Zeit, in der der Slave sein Antworttelegramm senden kann. Während des Wartens ist eine Kommunikation mit einem anderen Slave nicht möglich.

Zur Kommunikation mit mehreren Slaves ist für jeden Slave ein SEND-Datenbaustein für das Kommandotelegramm und ein RECEIVE-Datenbaustein für das Antworttelegramm erforderlich.

Eine Applikation mit mehreren Slaves würde aus entsprechend vielen Datenbausteinen mit Kommandos bestehen.

Diese werden der Reihe nach abgearbeitet:

1. Slave: Sende Kommandotelegramm an Slave-Adresse 1.Slave
   Empfange Antworttelegramm von Slave-Adresse 1.Slave
   Werte Antworttelegramm aus

2. Slave: Sende Kommandotelegramm an Slave-Adresse 2.Slave
   Empfange Antworttelegramm von Slave-Adresse 2.Slave
   Werte Antworttelegramm aus

... usw.

Eine Anforderung kann an einen bestimmten Slave gerichtet sein oder als Broadcast-Nachricht an alle Slaves gehen. Zur Kennzeichnung einer Broadcast-Nachricht wird die Slave-Adresse 0 eingetragen.

Nur Schreibaufträge dürfen als Broadcast gesendet werden.

Hinweis!

Nach einem Broadcast wartet der Master nicht auf ein Antworttelegramm.

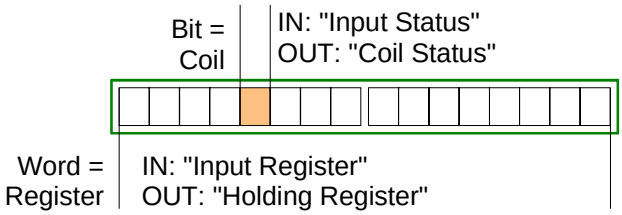
Master-Ausgabe-Bereich beschreiben

Durch "Ver-Oderung" des FC 0 Parameters *ANZ* mit 4000h werden zu sendende Slave-Daten nicht im Master-Eingabe- sondern im Master-Ausgabe-Bereich abgelegt. Da der Master unter Einsatz von Funktionscodes diesen Bereich lesen kann, können Sie diese Funktionalität beispielsweise zur direkten Fehlerübermittlung an den Master verwenden.

Modbus - Funktionscodes

Namens-konventionen

Für Modbus gibt es Namenskonventionen, die hier kurz aufgeführt sind:



- Modbus unterscheidet zwischen Bit- und Wortzugriff;
  Bits = "Coils" und Worte = "Register".
- Bit-Eingänge werden als "Input-Status" bezeichnet und Bit-Ausgänge als "Coil-Status".
- Wort-Eingänge werden als "Input-Register" und Wort-Ausgänge als "Holding-Register" bezeichnet.

Bereichs-definitionen

Üblicherweise erfolgt unter Modbus der Zugriff mittels der Bereiche 0x, 1x, 3x und 4x.

Mit 0x und 1x haben Sie Zugriff auf *digitale* Bit-Bereiche und mit 3x und 4x auf *analoge* Wort-Bereiche.

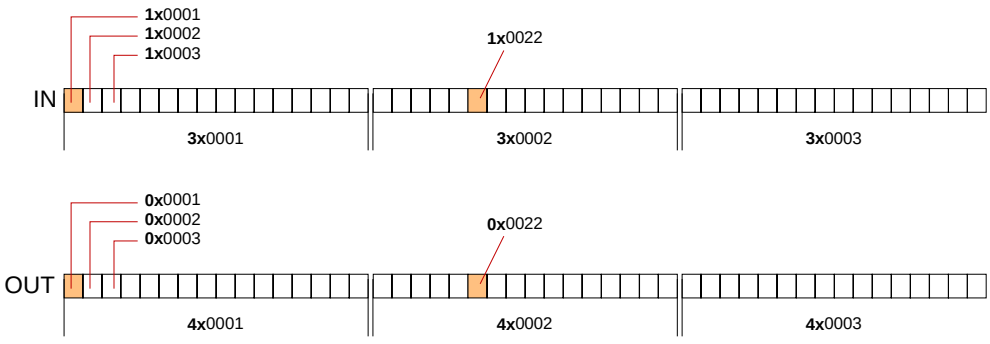
Da aber beim CP 240 von VIPA keine Unterscheidung zwischen Digital- und Analogdaten stattfindet, gilt folgende Zuordnung:

0x: Bit-Bereich für Ausgabe-Daten des Masters
Zugriff über Funktions-Code 01h, 05h, 0Fh

1x: Bit-Bereich für Eingabe-Daten des Masters
Zugriff über Funktions-Code 02h

3x: Wort-Bereich für Eingabe-Daten des Masters
Zugriff über Funktions-Code 04h

4x: Wort-Bereich für Ausgabe-Daten des Masters
Zugriff über Funktions-Code 03h, 06h, 10h



Eine Beschreibung der Funktions-Codes finden Sie auf den Folgeseiten.

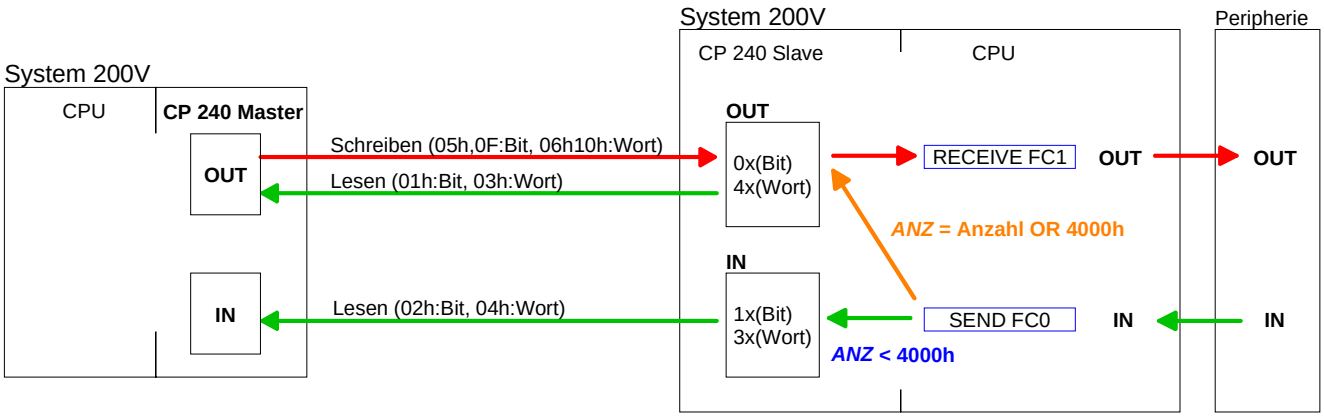
Übersicht

Mit folgenden Funktionscodes können Sie von einem Modbus-Master auf einen Slave zugreifen. Die Beschreibung erfolgt immer aus Sicht des Masters:

Code	Befehl	Beschreibung
01h	Read n Bits	n Bit lesen von Master-Ausgabe-Bereich 0x
02h	Read n Bits	n Bit lesen von Master-Eingabe-Bereich 1x
03h	Read n Words	n Worte lesen von Master-Ausgabe-Bereich 4x
04h	Read n Words	n Worte lesen von Master-Eingabe-Bereich 3x
05h	Write 1 Bit	1 Bit schreiben in Master-Ausgabe-Bereich 0x
06h	Write 1 Word	1 Wort schreiben in Master-Ausgabe-Bereich 4x
0Fh	Write n Bits	n Bit schreiben in Master-Ausgabe-Bereich 0x
10h	Write n Words	n Worte schreiben in Master-Ausgabe-Bereich 4x

Sichtweise für "Eingabe"- und "Ausgabe"-Daten

Die Beschreibung der Funktionscodes erfolgt immer aus Sicht des Masters. Hierbei werden Daten, die der Master an den Slave schickt, bis zu ihrem Ziel als "Ausgabe"-Daten (OUT) und umgekehrt Daten, die der Master vom Slave empfängt als "Eingabe"-Daten (IN) bezeichnet.



Antwort des Slaves

Liefert der Slave einen Fehler zurück, wird der Funktionscode mit 80h "verodert" zurückgesendet.

Ist kein Fehler aufgetreten, wird der Funktionscode zurückgeliefert.

Slave-Antwort:	Funktionscode OR 80h	→ Fehler
	Funktionscode	→ OK

Byte-Reihenfolge im Wort

Für die Byte-Reihenfolge im Wort gilt immer:

1 Wort

High-Byte	Low-Byte

Prüfsumme CRC, RTU, LRC

Die aufgezeigten Prüfsummen CRC bei RTU- und LRC bei ASCII-Modus werden automatisch an jedes Telegramm angehängt. Sie werden nicht im Datenbaustein angezeigt.

Read n Bits 01h, 02h

Code 01h: n Bit lesen von Master-Ausgabe-Bereich 0x
Code 02h: n Bit lesen von Master-Eingabe-Bereich 1x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Anzahl der gelesenen Bytes	Daten 1. Byte	Daten 2. Byte	...	Prüfsumme CRC/LRC
1Byte	1Byte	1Byte	1Byte	1Byte		1Wort
			max. 250Byte			

Read n Words 03h, 04h

03h: n Worte lesen von Master-Ausgabe-Bereich 4x
04h: n Worte lesen von Master-Eingabe-Bereich 3x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1.Bit	Anzahl der Worte	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Anzahl der gelesenen Bytes	Daten 1. Wort	Daten 2. Wort	...	Prüfsumme CRC/LRC
1Byte	1Byte	1Byte	1Wort	1Wort		1Wort
			max. 125Worte			

Write 1 Bit 05h

Code 05h: 1 Bit schreiben in Master-Ausgabe-Bereich 0x
Eine Zustandsänderung erfolgt unter "Zustand Bit" mit folgenden Werten:

"Zustand Bit" = 0000h → Bit = 0
"Zustand Bit" = FF00h → Bit = 1

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Write 1 Word 06h

Code 06h: 1 Wort schreiben in Master-Ausgabe-Bereich 4x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Write n Bits 0Fh

Code 0Fh: n Bit schreiben in Master-Ausgabe-Bereich 0x

Bitte beachten Sie, dass die Anzahl der Bits zusätzlich in Byte anzugeben sind.

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Anzahl der Bytes	Daten 1. Byte	Daten 2. Byte	...	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Byte	1 Byte	1 Byte	1 Byte	1 Wort
					max. 250 Byte			

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Wort

Write n Words 10h

Code 10h: n Worte schreiben in Master-Ausgabe-Bereich

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Wort	Anzahl der Worte	Anzahl der Bytes	Daten 1. Wort	Daten 2. Wort	...	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Byte	1 Wort	1 Wort	1 Wort	1 Wort
					max. 125 Worte			

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Wort	Anzahl der Worte	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Wort

Modbus - Fehlermeldungen

Übersicht

Bei der Kommunikation unter Modbus gibt es folgende 2 Fehlerarten:

- Master bekommt keine gültigen Daten
- Slave antwortet mit einer Fehlermeldung

Master bekommt keine gültigen Daten

Antwortet der Slave nicht innerhalb der vorgegebenen Wartezeit oder ist ein Telegramm fehlerbehaftet, trägt der Master im Empfangs-Baustein eine Fehlermeldung in Klartext ein.

Folgende Fehlermeldungen sind möglich:

ERROR01 NO DATA	*Error no data* Innerhalb der Wartezeit wurde kein Telegramm empfangen.
ERROR02 D LOST	*Error data lost* Es liegen keine Daten vor, da entweder der Empfangspuffer voll ist oder ein Fehler im Empfangsteil aufgetreten ist.
ERROR03 F OVERF	*Error frame overflow* Das Telegrammende wurde nicht erkannt und die maximale Telegrammlänge überschritten.
ERROR04 F INCOM	*Error frame incomplete* Es wurde nur ein Teiltelegramm empfangen.
ERROR05 F FAULT	*Error frame fault* Die Checksumme innerhalb eines Telegramms ist nicht korrekt.
ERROR06 F START	*Error frame start* Das Startzeichen ist falsch. Dieser Fehler kann ausschließlich unter Modbus-ASCII auftreten.

Slave antwortet mit einer Fehlermeldung

Liefert der Slave einen Fehler zurück, so wird der Funktionscode wie nachfolgend gezeigt mit 80h "verodert" zurückgesendet:

DB11.DBD 0	**DW#16#05900000**	**Antworttelegramm**
	mit 05 →	Slave-Adresse 05h
	90 →	Funktionscode 90h (Fehlermeldung, da 10h OR 80h = 90h)
	0000 →	Die Restdaten sind irrelevant, da Fehler rückgemeldet wurde.

Modbus - Beispiel

Übersicht

In dem nachfolgenden Beispiel wird eine Kommunikation zwischen einem Master und einem Slave über Modbus aufgebaut. Weiter soll das Beispiel zeigen, wie Sie unter Einsatz der Hantierungsbausteine auf einfache Weise die Kontrolle über die Kommunikationsvorgänge haben.

Bei Bedarf können Sie das Beispielprojekt von VIPA beziehen.

Voraussetzung

Folgende Komponenten sind für das Beispiel erforderlich:

- 2 System 200V bestehend aus CPU 21x mit CP 240
- Programmierkabel für MPI-Kopplung (z.B. Green Cable von VIPA)
- Serielles Verbindungskabel zur Verbindung beider CP 240

Vorgehensweise

- Bauen Sie ein Modbus-System bestehend aus Master-, Slave-System und verbinden Sie beide Systeme seriell.
- Projektieren Sie die Master-Seite! Öffnen Sie hierzu das Beispielprojekt in Ihrem Projektiertool. Passen Sie die Übertragungsparameter entsprechend an.
  Stellen Sie unter *Protokoll* "Modbus Master RTU" ein.
  Editieren Sie den OB1 und gleichen Sie ggf. die Modul-Adressen mit den Adressen der Parametrierung ab.
  Übertragen Sie Ihr Projekt in die CPU 21x der Master-Seite.
- Projektieren Sie die Slave Seite. Öffnen Sie hierzu das Beispielprojekt in Ihrem Projektiertool. Passen Sie in der Hardware-Konfiguration die CP 240 Parameter entsprechend an. Stellen Sie unter *Protokoll* "Modbus Slave RTU Short" ein. Geben Sie unter *Adresse* eine Slave-Adresse an. Für die Kommunikation unter Modbus ist auf Slave-Seite kein zusätzliches SPS-Programm erforderlich.

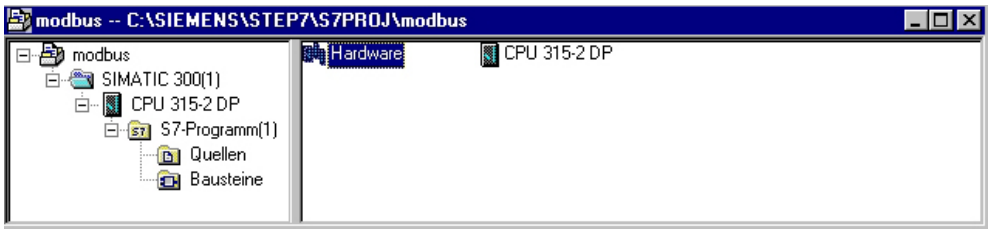
Projekt dearchivieren

Zum Dearchivieren in Ihr Konfigurationstool gehen Sie nach folgenden Schritten vor:

- Starten Sie den Siemens SIMATIC Manager.
- Zum Entpacken der Datei Modbus.zip gehen Sie auf **Datei** > *dearchivieren*.
- Wählen sie die Beispieldatei Modbus.zip aus und geben Sie als Zielverzeichnis "s7proj" an.
- Öffnen Sie nach dem Entpacken das Projekt.

Projekt-Struktur

Das Projekt hat folgende Struktur:



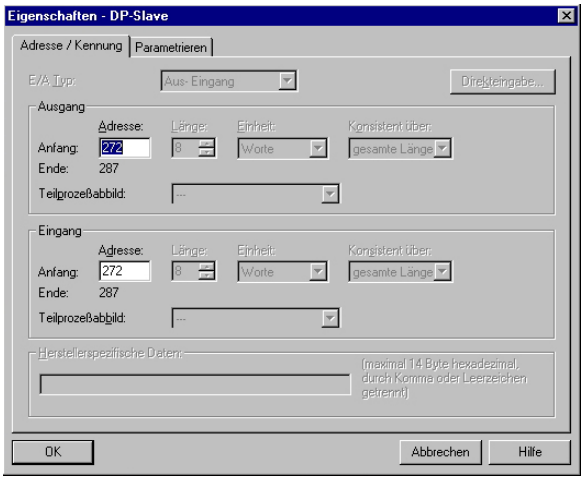
Master-Projektierung

Das Beispiel beinhaltet schon das SPS-Programm und die Parameter für den Modbus-Master. Sie müssen lediglich die Modbus-Parameter anpassen.

Parametrierung

Starten Sie hierzu den Hardware-Konfigurator und wählen Sie das Modul 240-1CA20 an. Durch Doppelklick gelangen Sie in die Parametrierung:

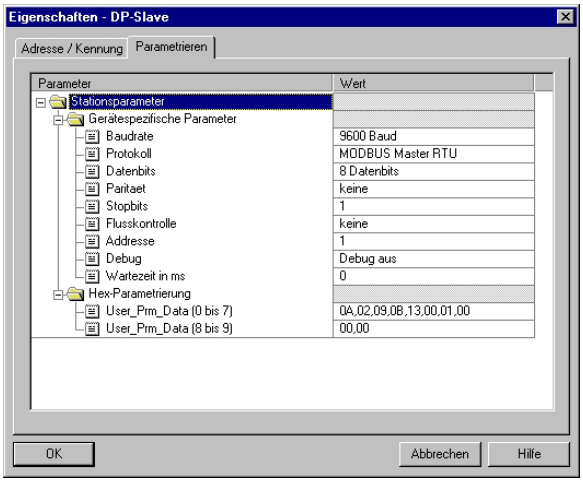
Dialog für Adress-Eingabe



Hier können Sie angeben, ab welcher Adresse die 16Byte für Ein- und Ausgabe in der CPU abliegen.

Bitte beachten Sie, dass Sie die Adressen, die Sie hier ändern, auch in Ihren SEND- und RECEIVE-Bausteinen ändern müssen.

Dialog für Modbus-Parameter



In diesem Teil der Parametrierung stellen Sie die Modbus-Parameter ein.

Folgende Parameter müssen bei allen Busteilnehmern gleich sein: Baudrate, Datenbits, Parität, Stopbits und Flusskontrolle.

Stellen Sie unter *Protokoll* "Modbus Master RTU" ein

Die Angabe einer Adresse ist nur auf der Slave-Seite erforderlich.

Bei der Master-Parametrierung wird die Adresse ignoriert.

SPS-Programm

Die gewünschten Modbus-Befehle geben Sie über Ihr SPS-Programm vor. Im vorliegenden Beispiel wird im OB1 der Einsatz von SEND und RECEIVE gezeigt.

```
OB 1:

    CALL     FC       0                  //"SEND"
      ADR           :=256                //Ausgangsadresse des Moduls
      _DB           :=DB10               //In diesem Datenbaustein erstellen
                                         //Sie das zu sendende Telegramm
      ABD           :=W#16#0             //Ab diesem Byte-Offset beginnt
                                         //das Telegramm im _DB
      ANZ           :=MW12               //Telegrammlaenge (zu sendende Laenge) in Byte
      PAFE          :=MB14               //Fehlerbyte
      FRG           :=M1.0               //Sendeanstoss (1=Anstoss, geht auf 0
                                         //wenn Senden abgeschlossen)
      GESE          :=MW16               //intern erforderlich
      ANZ_INT       :=MW18               //intern erforderlich
      ENDE_KOM      :=M2.0               //intern erforderlich
      LETZTER_BLOCK:=M2.1                //intern erforderlich
      SENDEN_LAEUFT:=M2.2                //intern erforderlich
      FEHLER_KOM    :=M2.3               //intern erforderlich

    CALL     FC       1                  //"RECEIVE"
      ADR           :=256                //Eingangsadresse des Moduls
      _DB           :=DB11               //In diesem Datenbaustein wird das
                                         //empfangene Telegramm abgelegt
      ABD           :=W#16#0             //Ab diesem Byte-Offset beginnt das Telegramm im _DB
      ANZ           :=MW22               //Telegrammlaenge (empfangene Laenge) in Byte
      EMFR          :=M1.1               //Empfangsstatus (1=Telegramm komplett empfangen)
      PAFE          :=MB34               //Fehlerbyte
      GEEM          :=MW36               //intern erforderlich
      ANZ_INT       :=MW38               //intern erforderlich
      EMPF_LAEUFT   :=M3.0               //intern erforderlich
      LETZTER_BLOCK:=M3.1                //intern erforderlich
      FEHL_EMPF     :=M3.2               //intern erforderlich

     U     M      1.1                    //solange Empfangsstatus=1 ist wird kein neues
                                         //Telegramm eingetragen
     R     M      1.1                    //daher muss der Empfangsstatus mit 0 quittiert werden
```

Passen Sie noch ggf. die Adressen, die der CP in der CPU belegt, an die Adressen in Ihrer Parametrierung an und übertragen Sie die Hardware-Konfiguration in Ihre CPU 21x des Master-Systems.

Hinweis!

Aufgrund der Übertragung der Daten in 8Byte-Blöcken, ist darauf zu achten, dass die Länge des Empfangsdatenbereichs ein Vielfaches von 8 ist. Auch sollten die schreibenden Master-Zugriffe nicht außerhalb des Empfangsbereichs liegen, da ansonsten der RECEIVE-Baustein einen Bereichsfehler meldet.

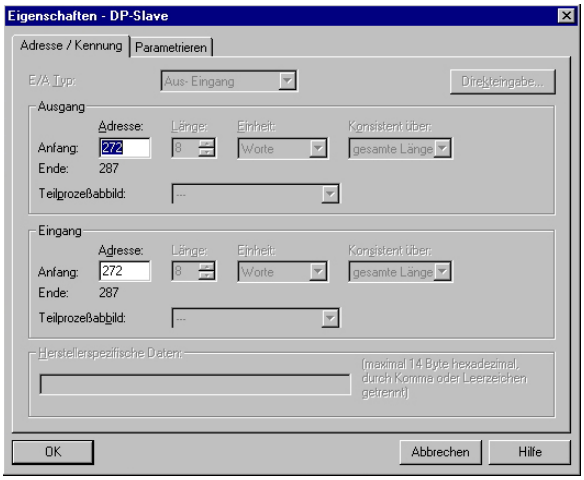
Slave-Projektierung

Für die Projektierung des Slave sind nur die Modbus-Parameter anzupassen. Ein SPS-Programm ist nicht erforderlich, da die Quell- und Zieldaten im Master-Telegramm mitgeliefert werden.

Parametrierung

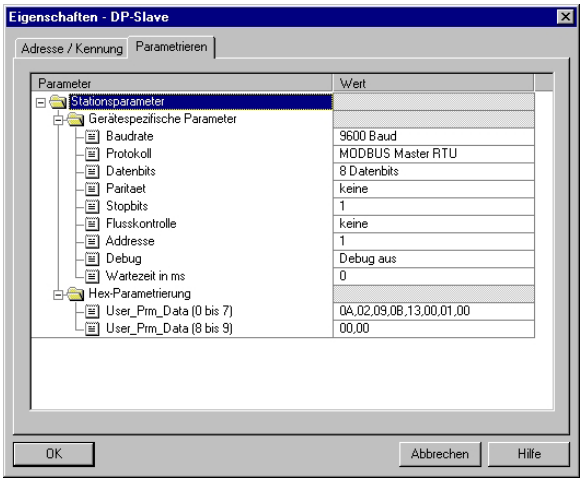
Zur Parametrierung des Slave-Moduls öffnen Sie das Beispielprojekt in Ihrem Hardware-Konfigurator. Wählen Sie das Modul 240-1CA20 an. Durch Doppelklick gelangen Sie in die Parametrierung.

Dialog für Adress-Eingabe



Hier können Sie angeben, ab welcher Adresse die 16Byte für Ein- und Ausgabe in der CPU abliegen.

Dialog für Modbus-Parameter



In diesem Teil der Parametrierung stellen Sie die Modbus-Parameter ein.

Folgende Parameter müssen bei allen Busteilnehmern gleich sein:

Baudrate, Datenbits, Parität, Stopbits und Flusskontrolle.

Geben Sie unter *Adresse* für den Slave eine gültige Modbus-Adresse an.

Übertragen Sie die Parametrierung in Ihre CPU des Slave-Systems.

Telegramme senden und empfangen

Öffnen Sie die Variablentabelle **Tabelle1** des Beispielprojekts und gehen Sie online.

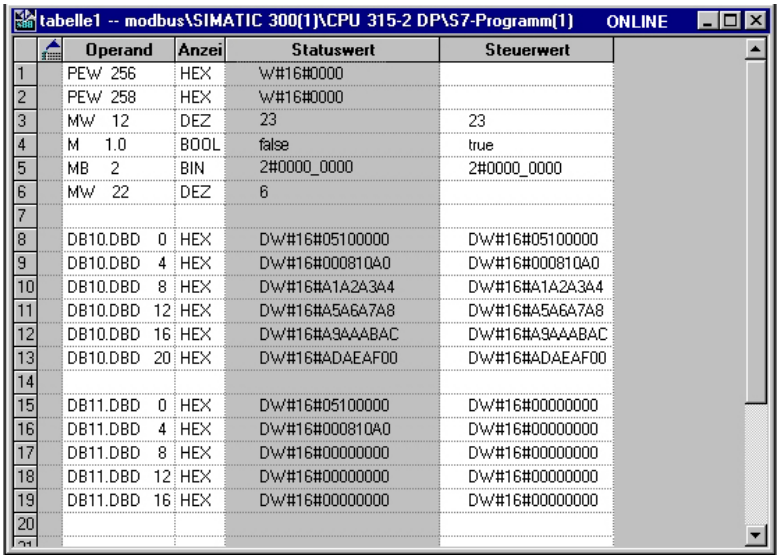


tabelle1 -- modbus\SIMATIC 300(1)\CPU 315-2 DP\S7-Programm(1) ONLINE

	Operand	Anzei	Statuswert	Steuerwert
1	PEW 256	HEX	W#16#0000	
2	PEW 258	HEX	W#16#0000	
3	MW 12	DEZ	23	23
4	M 1.0	BOOL	false	true
5	MB 2	BIN	2#0000_0000	2#0000_0000
6	MW 22	DEZ	6	
7				
8	DB10.DBD 0	HEX	DW#16#05100000	DW#16#05100000
9	DB10.DBD 4	HEX	DW#16#000810A0	DW#16#000810A0
10	DB10.DBD 8	HEX	DW#16#A1A2A3A4	DW#16#A1A2A3A4
11	DB10.DBD 12	HEX	DW#16#A5A6A7A8	DW#16#A5A6A7A8
12	DB10.DBD 16	HEX	DW#16#A9AAABAC	DW#16#A9AAABAC
13	DB10.DBD 20	HEX	DW#16#ADAEAF00	DW#16#ADAEAF00
14				
15	DB11.DBD 0	HEX	DW#16#05100000	DW#16#00000000
16	DB11.DBD 4	HEX	DW#16#000810A0	DW#16#00000000
17	DB11.DBD 8	HEX	DW#16#00000000	DW#16#00000000
18	DB11.DBD 12	HEX	DW#16#00000000	DW#16#00000000
19	DB11.DBD 16	HEX	DW#16#00000000	DW#16#00000000
20				

Sende-Baustein DB10

DB10.DBD 0	**DW#16#05100000** mit 05 → 10 → 0000 →	**Kommandotelegramm** Slave-Adresse 05h Funktionscode 10h (write n words) Offset 0000h
DB10.DBD 4	**DW#16#000810A0** mit 0008 → 10 → A0 →	**Kommandotelegramm + 1 Datenbyte** Wordcount 0008h Bytecount 10h Beginn 16 Byte Daten mit A0h
DB10.DBD 8	**DW#16#A1A2A3A4**	**Daten Byte 2 ... 5**
DB10.DBD 12	**DW#16#A5A6A7A8**	**Daten Byte 6 ... 9**
DB10.DBD 16	**DW#16#A9AAABAC**	**Daten Byte 10 ... 13**
DB10.DBD 20	**DW#16#ADAEAF00** mit ADAEAF → 00 →	**Daten Byte 14 ... 16 + 1 Byte nicht ben.** Daten Byte 14 ... 16 vom Modul nicht mehr belegt

Empfangs-Baustein DB11

DB11.DBD 0	**DW#16#05100000** mit 05 → 10 → 0000 →	**Antworttelegramm** Slave-Adresse 05h Funktionscode 10h (kein Fehler) Offset 0000h
DB11.DBD 4	**DW#16#000810A0** mit 0008 → 10 → A0 →	**Antworttelegramm + 1 Datenbyte** Wordcount 0008h Bytecount 10h Beginn 16 Byte Daten mit A0h (bei Schreibbefehl irrelevant)
DB11.DBD 8	**DW#16#00000000**	**Daten Byte 2 ... 5**
DB11.DBD 12	**DW#16#00000000**	**Daten Byte 6 ... 9**
DB11.DBD 16	**DW#16#00000000**	**Daten Byte 10 ... 13**
DB11.DBD 20	**DW#16#00000000** mit 000000 → 00 →	**Daten Byte 14 ... 16 + 1 Byte nicht ben.** Daten Byte 14 ... 16 vom Modul nicht mehr belegt

Empfangs-Baustein mit Fehlerrück-meldung

Slave antwortet nicht auf das Kommando des Masters

Antwortet der Slave nicht innerhalb der vorgegebenen Time-out-Zeit, trägt der Master im Empfangs-Baustein folgende Fehlermeldung ein:

ERROR01 NO DATA. In der Hex-Darstellung werden folgende Werte eingetragen:

DB11.DBD 0	**DW#16#4552524F**	**Antworttelegramm**
	mit 45 →	E
	52 →	R
	52 →	R
	4F →	O
DB11.DBD 4	**DW#16#52000120**	**Antworttelegramm**
	mit 52 →	R
	0001 →	0001h:1 (als Wort)
	20 →	" "
DB11.DBD 8	**DW#16#4E4F2044**	**Antworttelegramm**
	mit 4E →	N
	4F →	O
	20 →	" "
	44 →	D
DB11.DBD 12	**DW#16#41544100**	**Antworttelegramm**
	mit 41 →	A
	54 →	T
	41 →	A
	00 →	00h: (Nullterminierung)

.
.
.

Slave antwortet mit einer Fehlermeldung

Liefert der Slave einen Fehler zurück, so wird der Funktionscode mit 80h "verodert" zurückgesendet.

DB11.DBD	**DW#16#05900000**	**Antworttelegramm**
	mit 05 →	Slave-Adresse 05h
	90 →	Funktionscode 90h (Fehlermeldung, da 10h OR 80h = 90h)
	0000 →	Die Restdaten sind irrelevant, da Fehler rückgemeldet wurde.